6. Paketverwaltung und Updates

6.1 Paketverwaltung und Updates

Unter Linux wird Software meistens nicht einzeln von Webseiten heruntergeladen, sondern über eine Paketverwaltung installiert, aktualisiert und entfernt.

Die Paketverwaltung sorgt dafür, dass Programme, Bibliotheken, Abhängigkeiten und Updates kontrolliert verwaltet werden.

Für Fachinformatiker Systemintegration ist das wichtig, weil Server sicher, aktuell und nachvollziehbar administriert werden müssen.

Merksatz:

```
Paketverwaltung installiert,
aktualisiert
entfernt
und verwaltet Software kontrolliert.
```

Lernziele

Nach dieser Seite solltest du erklären können:

```
- was ein Paket ist
- was ein Paketmanager ist
- was ein Repository ist
- was Abhängigkeiten sind
- wie Updates grundsätzlich funktionieren
- warum Sicherheitsupdates wichtig sind
- was apt,
  dnf,
  zypper,
  pacman
  und apk sind
- wie man Software installiert,
  entfernt
  und aktualisiert
- welche typischen Fehler bei Paketverwaltung auftreten
```

Grundidee

Ein Linux-System besteht aus vielen Softwarepaketen.

Ein Paket enthält zum Beispiel:

- Programmdateien
- Bibliotheken
- Konfigurationsvorlagen
- Dokumentation
- Metadaten
- Installationsskripte
- Abhängigkeitsinformationen

Die Paketverwaltung weiß, welche Pakete installiert sind, welche Versionen verfügbar sind und welche Pakete voneinander abhängen.

Merksatz:

> Ein Paket ist eine verwaltete Softwareeinheit.

Paketmanager

Ein Paketmanager ist das Werkzeug, mit dem Software verwaltet wird.

Er kann:

- Pakete installieren
- Pakete entfernen
- Pakete aktualisieren
- Paketinformationen anzeigen
- Abhängigkeiten auflösen
- Paketquellen nutzen
- Versionsstände prüfen
- Sicherheitsupdates einspielen

Beispiele:

Distribution	Paketmanager
Debian	apt
Ubuntu	apt

Distribution	Paketmanager
Fedora	dnf
Rocky Linux	dnf
openSUSE	zypper
Arch Linux	pacman
Alpine Linux	apk

Merksatz:

```
Paketmanager unterscheiden sich je nach Distribution.
```

Repository

Ein Repository ist eine Paketquelle.

Dort liegen Softwarepakete, Metadaten und Versionsinformationen.

Ein System lädt aus Repositories Informationen darüber, welche Pakete verfügbar sind und welche Updates existieren.

Beispiele für Repository-Inhalte:

```
- Webserver
- Datenbanken
- Bibliotheken
- Sicherheitsupdates
- Systemwerkzeuge
- Entwicklungswerkzeuge
```

Merksatz:

```
Repository = Paketquelle.
```

Abhängigkeiten

Viele Programme benötigen andere Pakete, damit sie funktionieren.

Diese benötigten Pakete nennt man:

```
Abhängigkeiten
```

Beispiel:

```
Ein Programm benötigt eine bestimmte Bibliothek.
```

Die Paketverwaltung installiert diese Abhängigkeit automatisch mit, wenn sie fehlt.

Vorteil:

```
Programme werden nicht isoliert,
sondern passend zum System installiert.
```

Merksatz:

```
Abhängigkeiten sind Pakete,
die andere Pakete benötigen.
```

Warum Paketverwaltung statt manueller Installation?

Paketverwaltung hat viele Vorteile:

```
- zentrale Installation
- automatische Abhängigkeitsauflösung
- einfache Updates
- einfache Entfernung
- nachvollziehbare Paketlisten
- Sicherheitsupdates
- Versionsverwaltung
- saubere Integration ins System
```

Manuelle Installation kann problematisch sein, weil:

```
- Abhängigkeiten fehlen können
- Updates unklar sind
- Dateien schwer zu entfernen sind
- Sicherheitsupdates nicht automatisch kommen
```

- Versionskonflikte entstehen können

Merksatz:

Paketmanager sorgen für Ordnung,
Updates
und Nachvollziehbarkeit.

Debian und Ubuntu: apt

Debian und Ubuntu nutzen häufig:

```
apt
```

Typische Befehle:

Befehl	Bedeutung
apt update	Paketlisten aktualisieren
apt upgrade	installierte Pakete aktualisieren
apt install paket	Paket installieren
apt remove paket	Paket entfernen
apt purge paket	Paket inklusive Konfigurationsdateien entfernen
apt search begriff	Paket suchen
apt show paket	Paketinformationen anzeigen
apt autoremove	nicht mehr benötigte Abhängigkeiten entfernen

Merksatz:

apt update aktualisiert Paketlisten.
apt upgrade aktualisiert installierte Pakete.

---

apt update

Der Befehl:

```
apt update
```

lädt aktuelle Paketlisten aus den Repositories.

Wichtig:

```
apt update installiert noch keine Updates.
```

Es aktualisiert nur die Information darüber, welche Pakete und Versionen verfügbar sind.

Merksatz:

```
apt update aktualisiert die Paketinformationen,
nicht die Software selbst.
```

---

apt upgrade

Der Befehl:

```
apt upgrade
```

aktualisiert installierte Pakete, wenn neuere Versionen verfügbar sind.

Typische Nutzung:

```
apt update

apt upgrade
```

Bedeutung:

```
Erst Paketlisten aktualisieren,
dann Pakete aktualisieren.
```

Merksatz:

```
update holt Listen.
upgrade installiert Updates.
```

Paket installieren mit apt

Paket installieren:

```
apt install nginx
```

Bedeutung:

```
Das Paket nginx wird installiert.
Benötigte Abhängigkeiten werden automatisch mitinstalliert.
```

Häufig benötigt man administrative Rechte:

```
sudo apt install nginx
```

Merksatz:

```
apt install installiert Pakete.
```

Paket entfernen mit apt

Paket entfernen:

```
apt remove nginx
```

Das entfernt das Programm, aber Konfigurationsdateien können erhalten bleiben.

Paket inklusive Konfigurationsdateien entfernen:

```
apt purge nginx
```

Nicht mehr benötigte Abhängigkeiten entfernen:

```
apt autoremove
```

Merksatz:

```
remove entfernt Paket.
purge entfernt Paket plus Konfiguration.
```

Paket suchen mit apt

Paket suchen:

```
apt search nginx
```

Paketinformationen anzeigen:

```
apt show nginx
```

Installierte Pakete anzeigen:

```
apt list --installed
```

Merksatz:

```
apt search sucht Pakete.
apt show zeigt Details.
```

Fedora und Rocky Linux: dnf

Fedora und Rocky Linux nutzen häufig:

```
dnf
```

Typische Befehle:

Befehl	Bedeutung
dnf check-update	verfügbare Updates prüfen
dnf upgrade	Pakete aktualisieren
dnf install paket	Paket installieren
dnf remove paket	Paket entfernen
dnf search begriff	Paket suchen
dnf info paket	Paketinformationen anzeigen

Beispiele:

```
dnf install nginx

dnf upgrade

dnf remove nginx
```

Merksatz:

```
dnf ist typisch für Fedora-
und Enterprise-nahe Distributionen.
```

---

openSUSE: zypper

openSUSE nutzt häufig:

```
zypper
```

Typische Befehle:

Befehl	Bedeutung
zypper refresh	Paketquellen aktualisieren
zypper update	Pakete aktualisieren
zypper install paket	Paket installieren
zypper remove paket	Paket entfernen
zypper search begriff	Paket suchen
zypper info paket	Paketinformationen anzeigen

Merksatz:

```
zypper ist typisch für openSUSE.
```

Arch Linux: pacman

Arch Linux nutzt:

```
pacman
```

Typische Befehle:

Befehl	Bedeutung
pacman -Syu	System aktualisieren
pacman -S paket	Paket installieren
pacman -R paket	Paket entfernen
pacman -Ss begriff	Paket suchen
pacman -Qi paket	installiertes Paket anzeigen

Merksatz:

```
pacman ist typisch für Arch Linux.
```

Alpine Linux: apk

Alpine Linux nutzt:

```
apk
```

Typische Befehle:

Befehl	Bedeutung
apk update	Paketlisten aktualisieren
apk upgrade	Pakete aktualisieren
apk add paket	Paket installieren
apk del paket	Paket entfernen
apk search begriff	Paket suchen
apk info paket	Paketinformationen anzeigen

Alpine ist besonders häufig bei kleinen Container-Images.

Merksatz:

```
apk ist typisch für Alpine Linux.
```

Update, Upgrade und Dist-Upgrade

Die Begriffe können je nach Distribution unterschiedlich verwendet werden.

Bei Debian/Ubuntu gilt grob:

Befehl	Bedeutung
apt update	Paketlisten aktualisieren
apt upgrade	installierte Pakete aktualisieren
apt full-upgrade	Pakete aktualisieren, auch mit Entfernen/Ersetzen bei Bedarf

Wichtig:

```
full-upgrade kann stärker in das System eingreifen als upgrade.
```

Merksatz:

```
Nicht jedes Upgrade ist gleich tiefgreifend.
```

Sicherheitsupdates

Sicherheitsupdates schließen bekannte Schwachstellen.

Sie sind wichtig, weil ungepatchte Systeme ein Risiko darstellen.

Risiken ungepatchter Systeme:

```
- Ausnutzung bekannter Sicherheitslücken
- Malware-Infektion
- unberechtigter Zugriff
- Datenabfluss
- Manipulation
- Ausfall von Diensten
- Compliance-Probleme
```

Merksatz:

```
Patchmanagement reduziert bekannte Risiken.
```

Patchmanagement

Patchmanagement bedeutet:

```
Updates geplant,
kontrolliert
und nachvollziehbar einspielen.
```

Dazu gehören:

```
- Updates prüfen
- Risiko bewerten
- Wartungsfenster planen
- Backup erstellen
- Abhängigkeiten beachten
- Updates installieren
- Dienste prüfen
- Logs prüfen
- Rollback-Möglichkeit vorbereiten
- Dokumentation aktualisieren
```

Merksatz:

```
Updates sind nicht nur Technik,
sondern auch Prozess.
```

Warum Updates nicht blind einspielen?

Updates können Probleme verursachen.

Beispiele:

```
- Dienst startet nach Update nicht
- Konfiguration ist nicht mehr kompatibel
- Abhängigkeit ändert sich
- API-Verhalten ändert sich
- Kernel-Update benötigt Neustart
- Softwareversion passt nicht zur Anwendung
- Paket entfernt ältere Komponente
```

Deshalb wichtig:

```
testen

sichern

dokumentieren

Wartungsfenster nutzen
```

Merksatz:

```
Sicherheitsupdates sind wichtig,
aber produktive Systeme brauchen kontrollierte Änderungen.
```

Kernel-Updates

Kernel-Updates aktualisieren den Kern des Betriebssystems.

Wichtig:

```
Ein neuer Kernel wird häufig erst nach einem Neustart aktiv.
```

Prüfen:

```
uname -r
```

zeigt den aktuell laufenden Kernel.

Nach Kernel-Update:

```
Neustart planen

Bootfähigkeit prüfen

Dienste prüfen

Treiber oder Module beachten
```

Merksatz:

Kernel-Update braucht oft Neustart.

Distribution-Upgrade

Ein Distribution-Upgrade ist ein Wechsel auf eine neue Hauptversion der Distribution.

Beispiele:

Ubuntu 22.04 auf 24.04

Debian 12 auf Debian 13

Das ist umfangreicher als normale Paketupdates.

Vorher wichtig:

Backup

Kompatibilität prüfen

Release Notes lesen

Drittanbieter-Repositories prüfen

Wartungsfenster planen

Rollback-Plan

Testsystem nutzen

Merksatz:

Distribution-Upgrade ist größer als normales Update.

Paketquellen und Drittanbieter-Repositories

Neben offiziellen Repositories können Drittanbieter-Quellen eingebunden werden.

Vorteile:

neuere Software

spezielle Anwendungen

Herstellerpakete

Risiken:

Vertrauenswürdigkeit

Sicherheitsrisiko

Versionskonflikte

unklare Updateversorgung

Abhängigkeiten

Paketquelle kann wegfallen

Merksatz:

Drittanbieter-Repositories nur bewusst und dokumentiert nutzen.

GPG-Schlüssel und Paketvertrauen

Viele Paketquellen nutzen Signaturen, damit Pakete und Paketlisten geprüft werden können.

Grundidee:

Das System prüft,
ob Pakete aus einer vertrauenswürdigen Quelle stammen.

Wichtig:

Nur vertrauenswürdige Paketquellen und Schlüssel einbinden.

Risiko:

```
Eine unsichere Paketquelle kann manipulierte Software liefern.
```

Merksatz:

```
Paketquellen sind Vertrauensquellen.
```

Paketinformationen prüfen

Paketinformationen helfen, ein Paket vor der Installation einzuschätzen.

Bei apt:

```
apt show paket
```

Typische Informationen:

```
Version

Beschreibung

Abhängigkeiten

Paketquelle

Größe

Maintainer

Homepage
```

Merksatz:

```
Vor Installation prüfen,
was installiert wird.
```

Installierte Pakete anzeigen

Bei apt:

```
apt list --installed
```

Nach bestimmtem Paket suchen:

```
apt list --installed | grep nginx
```

Alternativen je nach Distribution:

```
dnf list installed

pacman -Q

apk info
```

Merksatz:

```
Installierte Pakete sollten nachvollziehbar sein.
```

Paketdateien und Konfiguration

Paketmanager installieren Dateien an bestimmte Orte.

Beispiele:

```
Programme nach /usr/bin

Konfigurationen nach /etc

Dienste nach systemd-Unit-Pfaden

Daten nach /var/lib

Logs nach /var/log
```

Wichtig:

```
Konfigurationsdateien können bei Updates besondere Behandlung bekommen.
Änderungen sollten dokumentiert werden.
```

Merksatz:

```
Pakete verteilen Dateien an standardisierte Orte.
```

Dienste nach Paketinstallation

Ein installiertes Paket bedeutet nicht immer, dass der Dienst läuft.

Nach Installation prüfen:

```
systemctl status dienst

systemctl is-enabled dienst

ss -tulpen
```

Beispiel:

```
nginx ist installiert,
aber der Dienst ist nicht gestartet.
```

Dann nötig:

```
systemctl start nginx

systemctl enable nginx
```

Merksatz:

```
Installiert heißt nicht automatisch:
Dienst läuft.
```

Paket entfernen und Reste

Beim Entfernen können Reste bleiben.

Mögliche Reste:

```
Konfigurationsdateien

Datenverzeichnisse

Logdateien

Benutzerkonten

systemd-Units

Cache-Dateien
```

Beispiel:

```
apt remove paket
```

entfernt meist nicht alle Konfigurationen.

```
apt purge paket
```

entfernt zusätzlich Paketkonfigurationen.

Wichtig:

```
Datenbanken oder Anwendungsdaten werden oft nicht automatisch gelöscht.
```

Merksatz:

```
Entfernen heißt nicht immer vollständig bereinigt.
```

Autoremove

autoremove entfernt Pakete, die als Abhängigkeiten installiert wurden und nicht mehr benötigt werden.

Bei apt:

```
apt autoremove
```

Wichtig:

```
Vorher prüfen,
welche Pakete entfernt werden sollen.
```

Merksatz:

```
autoremove räumt nicht mehr benötigte Abhängigkeiten auf.
```

Paketcache

Paketmanager speichern heruntergeladene Paketdaten teilweise im Cache.

Bei apt kann man aufräumen mit:

```
apt clean
```

oder:

```
apt autoclean
```

Nutzen:

```
Speicherplatz freigeben
```

Merksatz:

```
Paketcache kann Speicherplatz belegen.
```

Typische Fehler: Paketlisten veraltet

Fehlerbild:

```
Paket wird nicht gefunden
```

Mögliche Ursache:

```
Paketlisten sind veraltet.
```

Bei apt prüfen:

```
apt update
```

Danach erneut:

```
apt search paket

apt install paket
```

Merksatz:

```
Wenn apt ein Paket nicht findet,
zuerst Paketlisten prüfen.
```

Typische Fehler: Repository nicht erreichbar

Fehlerbild:

```
Paketmanager kann keine Paketlisten laden.
```

Mögliche Ursachen:

```
kein Internet

DNS-Problem

Proxy-Problem

Repository offline
```

falsche Paketquelle

Zertifikatsproblem

Firewall blockiert

Prüfen:

Netzwerkverbindung

DNS

Repository-URL

Proxy

Uhrzeit

Merksatz:

Paketfehler können Netzwerkfehler sein.

Typische Fehler: Abhängigkeitskonflikt

Fehlerbild:

Paket kann nicht installiert werden,
weil Abhängigkeiten nicht erfüllt sind.

Mögliche Ursachen:

gemischte Paketquellen

falsche Distribution-Version

Drittanbieter-Repository

```
Paketversion passt nicht

unvollständiges Upgrade
```

Merksatz:

```
Abhängigkeitsprobleme entstehen oft durch Versions-
oder Repository-Konflikte.
```

Typische Fehler: Paketdatenbank gesperrt

Fehlerbild bei apt:

```
Could not get lock
```

Bedeutung:

```
Ein anderer Paketverwaltungsprozess läuft gerade
oder wurde nicht sauber beendet.
```

Mögliche Ursachen:

```
automatisches Update läuft

anderer apt-Prozess offen

vorheriger Vorgang abgebrochen
```

Sicheres Vorgehen:

```
prüfen,
ob ein Paketprozess läuft

warten,
wenn Updates aktiv sind

nicht blind Sperrdateien löschen
```

Merksatz:

```
Paketdatenbank-Sperren nicht unüberlegt entfernen.
```

Typische Fehler: Speicherplatz voll

Paketinstallationen können fehlschlagen, wenn Speicherplatz fehlt.

Prüfen:

```
df -h

du -sh /var/cache

du -sh /var/log
```

Mögliche Maßnahmen:

```
Logs prüfen und sauber rotieren

Paketcache bereinigen

alte Kernel prüfen

nicht benötigte Pakete entfernen
```

Merksatz:

```
Updatefehler können Speicherplatzprobleme sein.
```

Typische Fehler: Dienst startet nach Update nicht

Mögliche Ursachen:

```
Konfigurationsänderung

veraltete Option
```

```
neue Version inkompatibel

Rechteproblem

Port bereits belegt

Abhängigkeit fehlt
```

Prüfen:

```
systemctl status dienst

journalctl -u dienst

Konfiguration prüfen

Änderungsnotizen prüfen

Rollback oder Fix planen
```

Merksatz:

```
Nach Update immer Dienste und Logs prüfen.
```

Paketverwaltung und Sicherheit

Paketverwaltung ist Teil der Systemsicherheit.

Sicheres Vorgehen:

```
- offizielle Paketquellen bevorzugen
- Drittanbieterquellen dokumentieren
- Updates regelmäßig prüfen
- Sicherheitsupdates priorisieren
- Dienste nach Updates testen
- Paketquellen absichern
- keine unbekannten Installationsskripte blind ausführen
- minimale Installation bevorzugen
```

Merksatz:

```
Weniger unnötige Pakete bedeutet kleinere Angriffsfläche.
```

Minimale Installation

Auf Servern sollte nur installiert sein, was benötigt wird.

Vorteile:

```
weniger Angriffsfläche

weniger Updates

weniger Abhängigkeiten

weniger Fehlerquellen

bessere Übersicht
```

Beispiel:

```
Ein Datenbankserver braucht normalerweise keine grafische Desktopumgebung.
```

Merksatz:

```
Installiere nur,
was für den Zweck nötig ist.
```

Paketverwaltung in Containern

Container-Images nutzen ebenfalls Paketmanager, aber anders als normale Server.

Beispiel Alpine:

```
apk add paket
```

Beispiel Debian-basiertes Image:

```
apt update

apt install paket
```

Wichtig bei Containern:

```
Images klein halten

Paketlisten nach Installation bereinigen

feste Versionen prüfen

keine unnötigen Tools installieren

Sicherheitsupdates über neue Images einspielen
```

Merksatz:

```
In Containern wird Software meist über neue Images aktualisiert.
```

Typische Paketmanager-Befehle im Vergleich

Aufgabe	Debian/Ubuntu	Fedora/Rocky	openSUSE	Arch	Alpine
Paketlisten aktualisieren	apt update	dnf check-update	zypper refresh	pacman -Sy	apk update
System aktualisieren	apt upgrade	dnf upgrade	zypper update	pacman -Syu	apk upgrade
Paket installieren	apt install	dnf install	zypper install	pacman -S	apk add
Paket entfernen	apt remove	dnf remove	zypper remove	pacman -R	apk del
Paket suchen	apt search	dnf search	zypper search	pacman -Ss	apk search
Paketinfo anzeigen	apt show	dnf info	zypper info	pacman -Qi	apk info

Merksatz:

```
Gleiche Aufgabe,
anderer Paketmanager.
```

Typische Admin-Reihenfolge bei Updates

Sicheres Vorgehen bei Servern:

```
1. System und Dienste dokumentieren

2. Backup oder Snapshot prüfen

3. freie Kapazität prüfen

4. Paketlisten aktualisieren

5. verfügbare Updates prüfen

6. Wartungsfenster beachten

7. Updates installieren

8. Dienste prüfen

9. Logs prüfen

10. Funktion testen

11. Dokumentation aktualisieren
```

Merksatz:

```
Updates brauchen Vorbereitung,
Durchführung
und Nachprüfung.
```

Typische Prüfungsfragen

Frage	Kurzantwort
Was ist ein Paket?	verwaltete Softwareeinheit
Was ist ein Paketmanager?	Werkzeug zur Verwaltung von Softwarepaketen
Was ist ein Repository?	Paketquelle

Frage	Kurzantwort
Was sind Abhängigkeiten?	benötigte Zusatzpakete
Was macht apt update?	Paketlisten aktualisieren
Was macht apt upgrade?	installierte Pakete aktualisieren
Was macht apt install?	Paket installieren
Was macht apt remove?	Paket entfernen
Was macht apt purge?	Paket inklusive Konfiguration entfernen
Was ist Patchmanagement?	geplanter Umgang mit Updates
Warum sind Sicherheitsupdates wichtig?	schließen bekannte Schwachstellen
Warum nicht blind updaten?	Dienste oder Abhängigkeiten können brechen
Warum minimale Installation?	kleinere Angriffsfläche

Typische Prüfungsfallen

Falle	Richtig denken
apt update als Softwareupdate verstehen	aktualisiert nur Paketlisten
apt upgrade vergessen	installiert die Updates
installierter Dienst läuft automatisch	nicht immer, systemctl prüfen
Paket entfernen heißt alles weg	Daten und Konfiguration können bleiben
Drittanbieter-Repos blind nutzen	Vertrauens- und Versionsrisiko
Updates ohne Backup	riskant bei produktiven Systemen
Kernel-Update ohne Neustart einplanen	neuer Kernel oft erst nach Reboot aktiv
Sicherheitsupdates aufschieben	bekannte Lücken bleiben offen
Paketfehler nur als Paketproblem sehen	Netzwerk, DNS, Speicher prüfen
manuelle Installationsskripte blind ausführen	Sicherheitsrisiko

---

IHK-sichere Kurzformulierung

Unter Linux wird Software meist über Paketmanager installiert, entfernt und aktualisiert. Ein Paket ist eine verwaltete Softwareeinheit, ein Repository ist eine Paketquelle. Paketmanager wie apt, dnf, zypper, pacman oder apk lösen Abhängigkeiten auf und stellen Updates bereit. Bei Debian und Ubuntu aktualisiert apt update die Paketlisten, während apt upgrade installierte Pakete aktualisiert. Sicherheitsupdates sind wichtig, weil sie bekannte Schwachstellen schließen. In produktiven Umgebungen sollten Updates geplant durchgeführt werden, mit Backup, Wartungsfenster, Prüfung der Dienste, Logkontrolle und Dokumentation. Drittanbieter-Repositories sollten nur bewusst genutzt werden, da sie Vertrauens- und Versionsrisiken mit sich bringen.

Merksätze

Paket = verwaltete Softwareeinheit.

Paketmanager verwaltet Pakete.

Repository = Paketquelle.

Abhängigkeiten sind benötigte Zusatzpakete.

apt ist typisch für Debian und Ubuntu.

dnf ist typisch für Fedora und Rocky Linux.

zypper ist typisch für openSUSE.

pacman ist typisch für Arch Linux.

apk ist typisch für Alpine Linux.

apt update aktualisiert Paketlisten.

apt upgrade aktualisiert installierte Pakete.

apt install installiert Pakete.

apt remove entfernt Pakete.

apt purge entfernt Paketkonfigurationen mit.

apt autoremove entfernt nicht mehr benötigte Abhängigkeiten.

Sicherheitsupdates schließen bekannte Lücken.

Patchmanagement ist ein Prozess.

Updates auf Servern brauchen Planung.

Kernel-Updates brauchen oft Neustart.

Installiert heißt nicht automatisch:
Dienst läuft.

Drittanbieter-Repositories sind Vertrauensquellen.

Paketfehler können Netzwerkprobleme sein.

Speicherplatzprobleme können Updates verhindern.

Minimale Installation reduziert Angriffsfläche.

Nach Updates Dienste,
Ports
und Logs prüfen.