ADMIN:DOCS

Fedora: Der umfassende Leitfaden zum Paketmanager DNF für Anfänger

Paketverwaltung unter Fedora und RHEL von Grund auf verstehen: Von RPM-Grundlagen und DNF5-Architektur über Repository-Management bis hin zu Transaktions-Rollbacks und Best Practices.

Kategorie: Linux-Neulinge **Veröffentlicht:** 20.09.2024 **Zuletzt geprüft:** 08.09.2026

Wer aus der Welt von Windows zu Fedora Linux wechselt, sucht im ersten Moment oft vergeblich nach vertrauten Mustern: Es gibt keine Installationsassistenten mit zahllosen „Weiter“-Buttons, keine unüberprüften `.exe`-Dateien auf zwielichtigen Download-Portalen und keine Treiber-CDs. Unter Linux läuft die Bereitstellung von Programmen fundamental anders – zentralisiert, kryptografisch abgesichert und vollautomatisch.

Das Herzstück dieses Systems auf Fedora, Red Hat Enterprise Linux (RHEL), AlmaLinux und Rocky Linux heißt **DNF** (**Dandified YUM**) bzw. in modernen Installationen **DNF5**.

Für Einsteiger wirkt ein Paketmanager im ersten Augenblick wie ein reiner App Store für die Kommandozeile. Doch DNF leistet weit mehr: Es wacht über ein engmaschiges Geflecht aus Zehntausenden Softwarekomponenten, löst hochkomplexe Abhängigkeiten mit mathematischer Präzision auf, schützt das Betriebssystem vor Versionskonflikten und protokolliert jede Transaktion so detailliert, dass du fehlerhafte Systemänderungen wie mit einer Zeitmaschine ungeschehen machen kannst.

Das Spektrum reicht vom grundlegenden RPM-Paketformat über den modernen C++-Standard DNF5 und Drittquellen wie RPM Fusion und Fedora COPR bis hin zu atomaren Transaktions-Rollbacks im laufenden Betrieb.

Quelle: https://admindocs.de/de/linux-neulinge/fedora-der-umfassende-leitfaden-vom-paketmanager-dnf

Die drei Säulen: Wie RPM, Repositories und DNF zusammenwirken

Um DNF sicher zu beherrschen, hilft ein Blick auf die drei Kernbausteine der Softwareverteilung in Fedora:



1. Das RPM-Format: Der Datencontainer

Das Format `.rpm` (*Red Hat Package Manager*) ist das elementare Paketarchiv. Eine `.rpm`-Datei enthält:

- Die eigentlichen Programmdateien, Bibliotheken und Konfigurationsdateien in komprimierter Form.
- Metadaten: Name, Versionsnummer, Lizenz, Architektur (z. B. `x86_64` oder `aarch64`) und Dateigröße.

- Steuer-Skripte: Aktionen, die vor oder nach der Installation ausgeführt werden müssen (z. B. das Anlegen eines Systembenutzers).
- Eine kryptografische Signatur (GPG), die sicherstellt, dass das Paket unverändert von den Fedora-Entwicklern stammt.
- Eine präzise Deklaration aller Abhängigkeiten (*„Ich benötige libssl.so.3 und glibc >= 2.38“*).

Das zugrundeliegende Werkzeug `rpm` kann solche Pakete zwar direkt auf der Festplatte entpacken und registrieren, stößt im Alltag jedoch sofort an Grenzen:

> □□ **Warum rpm allein nicht reicht:** Wenn ein Paket drei weitere Bibliotheken voraussetzt, bricht der Befehl `rpm -ivh programm.rpm` schlicht mit einer Fehlermeldung ab. Er weiß nicht, woher er die fehlenden Bausteine laden soll. Dieses historische Phänomen nannten Administratoren früher die gefürchtete *Dependency Hell*.

2. Repositories: Die kuratierten Softwarequellen

Ein **Repository** (Softwarequelle) ist ein strukturierter Webserver, auf dem tausende aufeinander abgestimmte RPM-Pakete gemeinsam mit einer maschinenlesbaren Inhaltsübersicht (den *Metadaten*) bereitgestellt werden. Fedora betreibt weltweit gespiegelte Servernetze (Mirrors), die sicherstellen, dass Downloads stets schnell und ausfallsicher geladen werden.

3. DNF: Der intelligente Lotse

Hier übernimmt **DNF** die Regie: Wenn du ein Programm anforderst, lädt DNF die Metadaten der Repositories, prüft die Abhängigkeiten aller beteiligten Komponenten, berechnet den optimalen Installationsplan, lädt alle benötigten RPMs herunter, prüft deren GPG-Schlüssel und übergibt die Pakete geordnet an `rpm` zur eigentlichen Installation.

Der Generationswechsel: Was DNF5 in Fedora auszeichnet

Mit Fedora 41 wurde der Schritt vollzogen, auf den Administratoren jahrelang hingearbeitet haben: **DNF5 hat den klassischen, Python-basierten Vorgänger DNF (DNF4) endgültig abgelöst.** Auf modernen Systemen verweist der Befehl `dnf` nahtlos auf die neue Binärdatei `/usr/bin/dnf5`. Auch in der Unternehmenswelt ist diese Architektur das neue Fundament: Sowohl CentOS Stream 10 als auch Red Hat Enterprise Linux 10 (RHEL 10) setzen standardmäßig auf diesen modernen C++-Stack.



Die wichtigsten Neuerungen und Unterschiede für Einsteiger im Überblick:

- **Pure C++-Performance:** DNF5 wurde von Grund auf in modernem C++ neu geschrieben und nutzt die Kernbibliothek `libdnf5`. Der Wegfall der Python-

Laufzeitumgebung führt zu bis zu viermal schnelleren Startzeiten und einem drastisch reduzierten Arbeitsspeicherbedarf.

- **Plattformübergreifender Standard (Enterprise-Ready):** Was in Fedora erprobt wurde, bildet in RHEL 10 und CentOS Stream 10 die gemeinsame technologische Basis. Das hier gelernte Wissen lässt sich nahtlos auf moderne Unternehmens-Server übertragen.
- **Gemeinsamer D-Bus-Daemon (`dnf5daemon`):** Desktop-Werkzeuge wie GNOME Software oder Cockpit greifen nun auf denselben Hintergrunddienst zu wie die Kommandozeile. Gesperrte Datenbanken oder divergierende Update-Stände gehören damit der Vergangenheit an.
- **Getrennte Historien-Datenbank:** DNF5 verwaltet seine Transaktionshistorie in einer eigenen SQLite-Datenbank unter `/var/lib/dnf5/history/`. Vorgänge, die noch mit dem alten DNF4 ausgeführt wurden, tauchen in der neuen Historie nicht mehr auf.
- **Wegfall von DeltaRPM:** Da moderne CPUs und Breitbandverbindungen schneller Daten übertragen als historische Delta-Diffs aufwendig dekomprimiert werden können, verzichtet DNF5 standardmäßig auf DeltaRPMs.

Repository-Architektur & Quellenverwaltung

Fedora unterteilt Softwarequellen in getrennte Bereiche, um Stabilität, Sicherheit und juristische Vorgaben (Open-Source-Lizenzen) transparent voneinander zu trennen.

Standard-Repositories unter Fedora

Jedes frisch installierte Fedora-System verfügt über drei vorkonfigurierte Hauptquellen:

- **fedora:** Das Basis-Repository mit dem eingefrorenen Softwarestand zum offiziellen Release-Tag der jeweiligen Fedora-Version.
- **updates:** Die kontinuierlich gepflegte Quelle für Fehlerbehebungen, Sicherheits-Patches und neue Kernel-Versionen.
- **updates-testing:** Vorab-Veröffentlichungen von Paketen für Tester. Standardmäßig deaktiviert, um Produktionssysteme nicht zu destabilisieren.

Tuning der Konfiguration in /etc/dnf/dnf.conf

Die zentrale Steuerdatei für den Paketmanager ist `/etc/dnf/dnf.conf`. Mit wenigen Anpassungen lässt sich das Verhalten im Alltag spürbar optimieren:

□□ Praktisches Beispiel:

Wir öffnen die Konfiguration mit Administratorrechten und passen zentrale Parameter an:

```
[main]
gpgcheck=True
installonly_limit=3
clean_requirements_on_remove=True
best=False
skip_if_unavailable=True
max_parallel_downloads=10
fastestmirror=True
```

Bedeutung der wichtigsten Direktiven:

- `max_parallel_downloads=10`: Lädt bis zu 10 RPM-Dateien gleichzeitig herunter. Auf schnellen Breitbandanschlüssen verkürzt dies System-Upgrades erheblich.
- `installonly_limit=3`: Bestimmt, wie viele Kernel-Versionen gleichzeitig im System verbleiben. So hast du im Boot-Menü (GRUB) stets zwei ältere, funktionierende Kernel zur Hand, falls ein Kernel-Update Probleme bereitet.
- `clean_requirements_on_remove=True`: Stellt sicher, dass beim Entfernen eines Programms auch alle Abhängigkeiten deinstalliert werden, die von keinem anderen Paket mehr benötigt werden.
- `fastestmirror=True`: Misst bei der Metadaten-Aktualisierung die Latenz der Server und wählt bevorzugt geografisch nahe Mirrors.

Repositories verwalten in DNF5

Unter DNF5 werden Repositories direkt über den integrierten Befehlszweig verwaltet. Das ersetzt die historischen Python-Plugins alter Versionen:

□□ Praktisches Beispiel:

Repositories auflisten, neue Quellen hinzufügen und aktivieren:

```
# Alle aktiven Repositories anzeigen
dnf repo list

# Auch deaktivierte Quellen anzeigen
dnf repo list --all

# Neues Repository aus einer Webquelle hinzufügen
sudo dnf config-manager addrepo --from-
repofile=https://download.docker.com/linux/fedora/docker-ce.repo

# Ein Repository gezielt aktivieren oder deaktivieren
sudo dnf config-manager enable docker-ce-stable
sudo dnf config-manager disable docker-ce-test
```

Unfreie Software & Multimedia: RPM Fusion einbinden

Das Fedora-Projekt verpflichtet sich strikt freier Software. Daher sind proprietäre Grafiktreiber (wie Nvidia) oder lizenzrechtlich geschützte Video-Codecs (z. B. H.264/H.265 für Videobearbeitung und Medienwiedergabe) nicht im Basissystem enthalten.

Die Fedora-Community stellt hierfür das externe Archiv **RPM Fusion** bereit, aufgeteilt in `free` (Open-Source-Software mit Patentrestriktionen) und `nonfree` (proprietäre Software wie Grafiktreiber):

□□ Praktisches Beispiel:

RPM Fusion Free und Nonfree mit einem Befehl im System aktivieren:

```
# RPM Fusion Repositories für die aktuelle Fedora-Version einbinden
sudo dnf install \
  https://mirrors.rpmfusion.org/free/fedora/rpmfusion-free-release-$(rpm -E
%fedora).noarch.rpm \
  https://mirrors.rpmfusion.org/nonfree/fedora/rpmfusion-nonfree-release-$(rpm -E
%fedora).noarch.rpm
```

Community-Software mit Fedora COPR

COPR (*Cool Other Package Repo*) ist das Fedora-Gegenstück zu Ubuntus PPAs oder Archs AUR. Entwickler und Community-Mitglieder können dort eigene Pakete bauen und bereitstellen, die noch nicht in die offiziellen Repositories aufgenommen wurden:

□□ Praktisches Beispiel:

Ein Community-Repository aus COPR durchsuchen, aktivieren und nutzen:

```
# Ein COPR-Repository für ein gewünschtes Tool aktivieren
sudo dnf copr enable atim/lazygit

# Das Paket wie gewohnt installieren
sudo dnf install lazygit

# Nicht mehr benötigtes COPR-Repo wieder deaktivieren
sudo dnf copr disable atim/lazygit
```

> ⚠ **Sicherheitshinweis zu COPR:** COPR-Repositories werden von einzelnen Personen betrieben und nicht vom Fedora-Sicherheitsteam überprüft. Aktiviere nur Quellen von

> Entwicklern, denen du vertraust.

Paketverwaltung im Alltag: Suchen, Installieren & Entfernen

Kommen wir zur täglichen Praxis auf der Konsole. DNF zeichnet sich durch intuitive Befehle aus, die sich leicht einprägen lassen.

1. Pakete suchen und analysieren

Bevor du Software installierst, möchtest du meist wissen, wie das Paket exakt heißt und was es beinhaltet:

□□ Praktisches Beispiel:

Pakete im Softwarekatalog finden und Details einsehen:

```
# Nach einem Suchbegriff in Namen und Beschreibungen suchen
dnf search webserver

# Ausführliche Metadaten zu einem Paket anzeigen (Version, Größe, Lizenz)
dnf info nginx
```

2. Der Retter in der Not: dnf provides

Ein Szenario, das jeder Linux-Admin kennt: Du folgst einer Anleitung und die Shell meldet `Befehl semanage nicht gefunden`. Du weißt aber nicht, in welchem RPM-Paket dieser Befehl steckt.

Hier hilft `dnf provides` (oder `dnf whatprovides`):

□□ Praktisches Beispiel:

Ermitteln, welches Paket eine bestimmte Datei oder einen Befehl bereitstellt:

```
# Welches Paket enthält den Befehl htop?
dnf provides /usr/bin/htop

# Welches Paket stellt das Werkzeug semanage bereit?
dnf provides "*bin/semanage"
```

DNF durchsucht die Dateilisten aller Repositories und meldet zuverlässig, dass du in diesem Fall `policycoreutils-python-utils` installieren musst.

3. Software installieren

Die Installation erfolgt mit dem Subkommando `install`. DNF berechnet vorab alle Abhängigkeiten und zeigt eine detaillierte Transaktionsübersicht:

□□ Praktisches Beispiel:

Pakete einzeln, gebündelt oder aus lokalen RPM-Dateien installieren:

```
# Einzelnes Paket installieren
sudo dnf install htop

# Mehrere Pakete in einer einzigen Transaktion installieren
sudo dnf install git tmux vim curl

# Eine manuell heruntergeladene lokale RPM-Datei installieren
sudo dnf install ./mein-programm-1.0.0.rpm
```

Beispielausgabe einer DNF-Transaktionszusammenfassung:

```
Updating and loading repositories:
Repositories loaded.
Package            Arch     Version         Repository      Size
Installing:
 htop              x86_64   3.3.0-2.fc41    fedora         1.8 M
Installing dependencies:
 libnl3            x86_64   3.9.0-1.fc41    fedora       340.0 k

Transaction Summary:
 Installing:         2 packages
 Total size:         2.1 M
 Total download size: 1.2 M
Is this ok [y/N]:
```

> **Nicht-interaktiver Modus für Skripte:** Wenn du Automatisierungs-Skripte schreibst oder die Abfrage nicht manuell mit `y` bestätigen möchtest, nutze den Schalter `-y`: `sudo dnf install -y htop`.

4. Software deinstallieren und System sauber halten

Das Entfernen von Paketen erfolgt über `remove`:

Praktisches Beispiel:

Programme deinstallieren und verwaiste Bibliotheken aufspüren:

```
# Ein Programm entfernen
sudo dnf remove htop

# Automatische Bereinigung aller ungenutzten, verwaisten Abhängigkeiten
```

```
sudo dnf autoremove
```

`dnf autoremove` ist essenziell für die Systemhygiene: Es entfernt Bibliotheken, die einst als Abhängigkeit für ein Programm installiert wurden, nun aber von keiner einzigen installierten Software mehr gebraucht werden.

5. Paketgruppen verwalten

Möchtest du eine vollständige Arbeitsumgebung aufsetzen (beispielsweise zum Kompilieren von Software aus Quellcode), musst du nicht dutzende Einzelpakete wie `gcc`, `make`, `autoconf` und `gdb` manuell zusammensuchen:

□□ Praktisches Beispiel:

Arbeiten mit standardisierten Fedora-Paketgruppen:

```
# Alle verfügbaren Paketgruppen auflisten
dnf group list

# Die gesamte C/C++ Entwicklungsumgebung installieren
sudo dnf group install "Development Tools"

# Eine Gruppe wieder entfernen
sudo dnf group remove "Development Tools"
```

Systemaktualisierung & Release-Upgrades

Ein aktuelles System ist das wichtigste Fundament für IT-Sicherheit. DNF trennt zwischen regulären Updates, isolierten Sicherheitsaktualisierungen und Versions-Upgrades.

Reguläre Updates einspielen

□□ Praktisches Beispiel:

Prüfen und Installieren verfügbarer Systemaktualisierungen:

```
# Prüfen, welche Pakete aktualisiert werden können (ohne Installation)
dnf check-update

# Alle installierten Pakete auf den neuesten verfügbaren Stand bringen
sudo dnf upgrade
```

Gezielte Sicherheitsupdates einspielen

In Produktivumgebungen oder auf Servern möchte man oft nicht das gesamte System aktualisieren, sondern ausschließlich bekannte Schwachstellen (CVEs) schließen. In DNF5 wird dieses Management über das Subkommando `advisory` gesteuert (in früheren DNF4-Versionen als `updateinfo` bekannt, welches weiterhin als transparenter Kompatibilitäts-Alias unterstützt wird):

□□ Praktisches Beispiel:

Sicherheits-Advisories prüfen und gezielt einspielen:

```
# Zusammenfassung aller gemeldeten Sicherheitslücken anzeigen (DNF5 Standard)
dnf advisory summary

# Detaillierte Liste mit CVE-Nummern und Schweregrad anzeigen
dnf advisory list --security

# Alternativ über den bekannten Kompatibilitäts-Alias aus älteren DNF-Versionen
```

```
dnf updateinfo summary

# Ausschließlich Pakete aktualisieren, die Sicherheitslücken schließen
sudo dnf upgrade --security
```

Upgrade auf eine neue Fedora-Hauptversion

Fedora veröffentlicht alle sechs Monate eine neue Hauptversion. Der Umstieg (z. B. von Fedora 41 auf Fedora 42) erfolgt direkt im laufenden Betrieb über das offizielle `system-upgrade`-Verfahren:

□□ Praktisches Beispiel:

Vollständiges Distributions-Upgrade im fehlersicheren Offline-Modus:

```
# 1. Bestehendes System vollständig aktualisieren und Metadaten auffrischen
sudo dnf upgrade --refresh

# 2. Alle Pakete für die neue Zielversion im Hintergrund herunterladen
sudo dnf system-upgrade download --releasever=42

# 3. Den Upgrade-Vorgang im isolierten Wartungs-Reboot ausführen
sudo dnf system-upgrade reboot
```

Der eigentliche Installationsprozess findet während des Neustarts in einer minimalen Umgebung statt. Dadurch wird verhindert, dass laufende Dienste oder Desktop-Sitzungen während des Austauschs zentraler Systembibliotheken abstürzen.

Die Zeitmaschine: Transaktions-Historie und Rollbacks

Das herausragende Alleinstellungsmerkmal von DNF gegenüber Werkzeugen wie APT ist die lückenlose **Transaktions-Historie**. Jede Installation, jedes Update und jede Deinstallation wird

atomar in einer SQLite-Datenbank protokolliert.



Transaktionen analysieren und rückgängig machen

Stell dir vor, du hast ein Programm installiert, das dein System instabil macht oder Abhängigkeiten verändert hat. Anstatt mühsam zu rekonstruieren, welche Bibliotheken mitinstalliert wurden, nutzt du die Historie:

□□ Praktisches Beispiel:

Transaktionsverlauf einsehen, Details prüfen und Transaktionen rückgängig machen:

```
# Liste aller vergangenen DNF-Aktionen anzeigen
dnf history list

# Detailliert anzeigen, was in Transaktion 42 passiert ist
dnf history info 42

# Genau diese eine Transaktion rückgängig machen
sudo dnf history undo 42

# Das System auf den Zustand VOR einer bestimmten Transaktion zurücksetzen
sudo dnf history rollback 41
```

> ⚠ **Grenzen des Rollbacks:** Ein `history rollback` kann nur dann erfolgreich sein, wenn alle älteren RPM-Pakete, die für den früheren Zustand benötigt werden, noch in den Repositories oder im lokalen Cache existieren. Auf Desktop-Systemen mit Fedora ist `undo` für einzelne Transaktionen die sicherere und verlässlichere Wahl.

Modernes Fedora: Wann DNF, wann Flatpak, wann Toolbx?

Auf einer modernen Fedora Workstation existiert DNF nicht isoliert. Um dein System stabil zu halten, solltest du die Aufgabenteilung kennen:

Einsatzzweck	Empfohlenes Werkzeug	Warum?
Systemnahe Tools & CLI	**DNF / DNF5**	Shell-Tools (`tmux` , `git` , `htop`), Serverdienste und Systembibliotheken gehören direkt ins Basissystem.

Einsatzzweck	Empfohlenes Werkzeug	Warum?
Grafische Desktop-Apps	**Flatpak (Flathub)**	Browser, Messenger, Bildbearbeitung und Office laufen isoliert in einer Sandbox und gefährden keine System-Libs.
Entwicklungsumgebungen	**Toolbx / Distrobox**	Halte Compiler, Node.js-Versionen oder Python-Pakete in OCI-Containern, um dein Basissystem schlank zu halten.

Fehlersuche & Store-Hygiene (Troubleshooting)

Auch der beste Paketmanager benötigt gelegentlich administrative Zuwendung. Hier sind die vier wichtigsten Troubleshooting-Handgriffe:

1. Beschädigten Metadaten-Cache bereinigen

Wenn DNF meldet, dass Metadaten korrupt sind oder Checksummen nicht übereinstimmen:

□□ Praktisches Beispiel:

Lokalen Cache leeren und Metadaten frisch von den Servern spiegeln:

```
# Den gesamten lokalen Paket- und Metadaten-Cache leeren
sudo dnf clean all

# Cache komplett neu aufbauen
sudo dnf check-update
```

2. GPG-Schlüsselvalidierung reparieren

Verweigert DNF die Installation eines Pakets mit dem Hinweis auf einen nicht vertrauenswürdigen oder abgelaufenen GPG-Schlüssel, kannst du die offiziellen Schlüssel manuell importieren:

□□ Praktisches Beispiel:

Offizielle Fedora-Signaturschlüssel neu einlesen:

```
# Offizielle Fedora-Release-Keys importieren
sudo rpm --import /etc/pki/rpm-gpg/RPM-GPG-KEY-fedora-$(rpm -E %fedora)-primary
```

3. Pakete vor Updates sperren (Versionlock)

Möchtest du verhindern, dass ein geschäftskritischer Dienst oder ein spezifischer Treiber bei einem allgemeinen `dnf upgrade` aktualisiert wird, kannst du Pakete sperren. In DNF5 ist das Versionlock-Kommando nativ im Paket `dnf5-plugins` enthalten und speichert Ausnahmen strukturiert im TOML-Format unter `/etc/dnf/versionlock.toml`:

□□ Praktisches Beispiel:

Pakete mit Versionlock fixieren:

```
# Versionlock-Plugin installieren (falls noch nicht vorhanden)
sudo dnf install dnf5-plugins

# Paket auf aktueller Version festnageln
sudo dnf versionlock add nginx

# Alle gesperrten Pakete auflisten (liest /etc/dnf/versionlock.toml)
dnf versionlock list

# Sperre wieder aufheben
sudo dnf versionlock delete nginx
```

4. Inkonsistenzen der RPM-Datenbank reparieren

Sollte das System während einer Transaktion abgestürzt sein und die Paketdatenbank blockieren:

□□ Praktisches Beispiel:

Paketdatenbank auf Beschädigungen prüfen und reparieren:

```
# Auf unvollständige oder duplizierte Pakete prüfen
sudo dnf check

# Die interne RPM-Datenbank neu indizieren
sudo rpm --rebuilddb
```

Befehlsreferenz (Cheatsheet)

Kategorie	Befehl	Funktion & Zweck
Suchen	`dnf search <begriff>`	Durchsucht Paketnamen und Zusammenfassungen
Info	`dnf info <paket>`	Zeigt detaillierte Metadaten zu Version und Lizenz
Dateisuche	`dnf provides <pfad>`	Ermittelt, welches Paket eine bestimmte Datei liefert
Installation	`sudo dnf install <paket>`	Installiert Pakete inklusive aller Abhängigkeiten
Deinstallation	`sudo dnf remove <paket>`	Deinstalliert ein Paket aus dem System
Bereinigung	`sudo dnf autoremove`	Entfernt verwaiste, ungenutzte Abhängigkeiten
Aktualisierung	`sudo dnf upgrade`	Aktualisiert alle installierten Softwarepakete
Advisories	`dnf advisory summary`	Listet Sicherheitsmeldungen & CVEs (Alias: `updateinfo`)

Kategorie	Befehl	Funktion & Zweck
Sicherheit	`sudo dnf upgrade --security`	Spielt gezielt nur Sicherheits-Patches ein
Versionlock	`sudo dnf versionlock add <pkg>`	Fixiert Paketversion in `/etc/dnf/versionlock.toml`
Gruppen	`sudo dnf group install <gruppe>`	Installiert vorkonfigurierte Paketbündel
Historie	`dnf history list`	Zeigt alle vergangenen Transaktionen mit ID an
Rückgängig	`sudo dnf history undo <id>`	Macht eine Transaktion gezielt ungeschehen
Quellen	`dnf repo list`	Listet alle aktiven Repositories auf
Cache	`sudo dnf clean all`	Bereinigt lokale Metadaten- und Download-Caches

Weiterführende Ressourcen

Ressource	Beschreibung	Typ
Offizielle Fedora Dokumentation	Umfassende Dokumentation für Fedora Workstation, Server und CoreOS	Offizielle Dokumentation
DNF5 Projekt-Dokumentation	Offizielle C++-Architekturreferenz und Handbuch aller DNF5-Befehle	Handbuch & Referenz
RPM Fusion Portal	Offizielle Anleitungen für Multimedia-Codecs und Nvidia-Grafiktreiber	Community-Portal
Fedora COPR Build-System	Zentrales Verzeichnis von Community-Repositories und Builds	Community-Plattform
Fedora Packages Explorer	Interaktive Paketsuche mit Changelogs, Abhängigkeiten und Versionen	Offizielle Paketsuche

Fazit

Mit DNF und der modernen DNF5-Architektur verfügt Fedora über eines der fortschrittlichsten Paketmanagementsysteme der gesamten Unix-Welt. Wo andere Distributionen bei Paketkonflikten oft manuelle Eingriffe erfordern, glänzt DNF durch den mathematischen SAT-Solver (`libsolv`), extrem schnelle Metadatenverarbeitung in nativem C++ und eine lückenlose Transaktionshistorie.

Wer die Unterscheidung zwischen RPM-Dateien und Repositories verinnerlicht hat, mit `dnf provides` fehlende Befehle zielsicher aufspürt und die Historie mit `undo` als persönliches Sicherheitsnetz begreift, steuert sein System souverän und stressfrei.

> **Praxis-Tipp für deinen Alltag:** Trage dir in der Datei `/etc/dnf/dnf.conf` die Option `max_parallel_downloads=10` ein. Bei größeren System-Upgrades oder der Einrichtung einer neuen Maschine lädt DNF die Pakete parallel herunter, was die Wartezeit auf schnellen Internetanschlüssen drastisch verkürzt.