apt i dpkg: odpowiedniki na Debianie i Ubuntu

Poziom: podstawy **Dotyczy: Debian/Ubuntu** **Czas: ~20 min**

Debian i Ubuntu pakują oprogramowanie inaczej niż RHEL/Rocky: format `.deb` zamiast `.rpm`, `dpkg` jako warstwa niskopoziomowa i `apt` jako menedżer wysokopoziomowy, który dokłada rozwiązywanie zależności i pobieranie z sieci. Ta strona to słownik przekładu dla kogoś, kto zna `dnf`, i komplet fiszek dla codziennej pracy na Ubuntu albo Debianie.

Skróty użyte na tej stronie (2)

Skrót	Co znaczy
GPG	GNU Privacy Guard: program do podpisywania i szyfrowania plików oraz sprawdzania podpisów paczek
RHEL	Red Hat Enterprise Linux: komercyjna dystrybucja Linuksa

Fiszka

TYLKO ODCZYT

Tabela przekładu: dnf → apt → zypper

Trzy rodziny dystrybucji, trzy menedżery pakietów, ten sam zestaw codziennych czynności. Tabela poniżej jest głównym punktem odniesienia tej strony — reszta Fiszki rozbija kolumnę środkową na pojedyncze komendy z przykładami.

Czynność	`dnf` (RHEL/Rocky)	`apt` (Debian/Ubuntu)	`zypper` (SLES/openSUSE)
Odświeżenie metadanych repozytoriów	`dnf check-update`	`apt update`	`zypper refresh`
Aktualizacja pakietów, bez usuwania niczego	`dnf upgrade`	`apt upgrade`	`zypper update`

Czynność	`dnf` (RHEL/Rocky)	`apt` (Debian/Ubuntu)	`zypper` (SLES/openSUSE)
Aktualizacja, która może usunąć/podmienić pakiety	`dnf distro-sync`	`apt full-upgrade`	`zypper dist-upgrade`
Instalacja pakietu	`dnf install nazwa`	`apt install nazwa`	`zypper install nazwa`
Usunięcie pakietu, plik konfiguracyjny zostaje	`dnf remove nazwa`	`apt remove nazwa`	`zypper remove nazwa`
Usunięcie pakietu razem z konfiguracją	brak osobnej komendy (rpm i tak kasuje `%config` ostrożnie)	`apt purge nazwa`	brak osobnej komendy
Usunięcie osieroconych zależności	`dnf autoremove`	`apt autoremove`	`zypper packages --unneeded` (ręczne usunięcie)
Wyszukanie pakietu po nazwie/opisie	`dnf search fraza`	`apt search fraza`	`zypper search fraza`
Szczegóły pakietu	`dnf info nazwa`	`apt show nazwa`	`zypper info nazwa`
Lista zainstalowanych pakietów	`dnf list installed`	`apt list --installed`	`zypper packages --installed-only`
Lista pakietów do aktualizacji	`dnf list updates`	`apt list --upgradable`	`zypper list-updates`
Zablokowanie wersji pakietu	`dnf versionlock add nazwa`	`apt-mark hold nazwa`	`zypper addlock nazwa`
Zdjęcie blokady	`dnf versionlock delete nazwa`	`apt-mark unhold nazwa`	`zypper removelock nazwa`
Który pakiet dostarcza dany plik	`dnf provides /sciezka/plik`	`apt-file search /sciezka/plik`	`zypper what-provides /sciezka/plik`
Zależności pakietu	`dnf repoquery --requires nazwa`	`apt-cache depends nazwa`	`zypper info --requires nazwa`
Lista wszystkich plików z pakietu	`rpm -ql nazwa`	`dpkg -L nazwa`	`rpm -ql nazwa`
Do jakiego pakietu należy plik na dysku	`rpm -qf /sciezka`	`dpkg -S /sciezka`	`rpm -qf /sciezka`
Instalacja z lokalnego pliku pakietu	`dnf install ./plik.rpm`	`dpkg -i plik.deb`	`zypper install ./plik.rpm`
Weryfikacja plików zainstalowanego pakietu	`rpm -V nazwa`	`dpkg -V nazwa`	`rpm -V nazwa`
Dodanie repozytorium	`dnf config-manager --add-repo url`	`add-apt-repository ppa:...`	`zypper ar url alias`
Naprawa przerwanej instalacji	`dnf history undo` / ponowne `dnf install`	`dpkg --configure -a`	`zypper install --force-resolution`
Log operacji pakietowych	`/var/log/dnf.log`	`/var/log/apt/history.log`	`/var/log/zypp/history`

Trzy szczegóły, które tabela spłaszcza, ale które warto znać zanim pójdziemy dalej: `apt upgrade` nie ma prawa nic usunąć, więc czasem świadomie *nie* zainstaluje aktualizacji, jeśli wymagałaby ona usunięcia pakietu — o tym w sekcji *Krok po kroku*. `remove` i `purge` różnią się plikami konfiguracyjnymi, nie samym pakietem. I `apt` jako komenda dla człowieka ma inny status stabilności niż `apt-get` w skryptach — też niżej.

RHEL = Red Hat Enterprise Linux: komercyjna dystrybucja Linuksa · **SLES** = SUSE Linux Enterprise Server: płatna dystrybucja Linuksa firmy SUSE, konkurent Rocky/RHEL

ZMIENIA SYSTEM

Odświeżenie metadanych repozytoriów: `apt update`

```
sudo apt update
```

Ściągamy listy pakietów i metadane repozytoriów z `/etc/apt/sources.list` i `/etc/apt/sources.list.d/`, niczego nie instalując. Odpowiednik `dnf check-update`.

Przykład:

```
Get:2 http://archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Reading package lists... Done
5 packages can be upgraded. Run 'apt list --upgradable' to see them.
```

apt = Advanced Package Tool: menedżer pakietów Debiana i Ubuntu

ZMIENIA SYSTEM

Aktualizacja bez usuwania pakietów: `apt upgrade`

```
sudo apt upgrade
```

Podnosimy wersje pakietów do najnowszych dostępnych, ale **nigdy nie usuwamy** zainstalowanego pakietu ani nie instalujemy nowego — jeśli aktualizacja tego wymaga, pakiet zostaje w starej wersji.

Przykład:

```
The following packages have been kept back:
  linux-generic linux-headers-generic
0 upgraded, 0 newly installed, 0 to remove and 2 not upgraded.
```

apt = Advanced Package Tool: menedżer pakietów Debiana i Ubuntu

ZMIENIA SYSTEM

Aktualizacja, która może usuwać i podmieniać pakiety: `apt full-upgrade`

```
sudo apt full-upgrade
```

To samo co `upgrade`, ale bez tego ograniczenia — usuwamy albo podmieniamy pakiet, jeśli tego wymaga nowa wersja. Odpowiednik `dnf distro-sync`.

Przykład:

```
The following packages will be REMOVED:
  linux-image-6.8.0-31-generic
The following NEW packages will be installed:
  linux-image-6.8.0-45-generic
```

apt = Advanced Package Tool: menedżer pakietów Debiana i Ubuntu

ZMIENIA SYSTEM

Instalacja pakietu: `apt install`

```
sudo apt install NazwaPakietu
```

Instalujemy pakiet razem z brakującymi zależnościami; kilka nazw naraz trafia do jednej transakcji.

Przykład:

```
The following NEW packages will be installed:
  nginx nginx-common nginx-core
Setting up nginx (1.24.0-2ubuntu7) ...
```

apt = Advanced Package Tool: menedżer pakietów Debiana i Ubuntu

TYLKO ODCZYT

Usunięcie pakietu z zachowaniem konfiguracji: `apt remove`

```
sudo apt remove NazwaPakietu
```

Usuwamy binarki i pliki programu, ale **zostawiamy pliki konfiguracyjne** w `/etc` na wypadek ponownej instalacji.

Przykład:

```
Removing nginx (1.24.0-2ubuntu7) ...
```

apt = Advanced Package Tool: menedżer pakietów Debiana i Ubuntu

TYLKO ODCZYT

Usunięcie pakietu razem z konfiguracją: `apt purge`

```
sudo apt purge NazwaPakietu
```

To samo co `remove`, plus kasowanie plików konfiguracyjnych w `/etc`.

Przykład:

```
Purging configuration files for nginx-common (1.24.0-2ubuntu7) ...
```

apt = Advanced Package Tool: menedżer pakietów Debiana i Ubuntu

TYLKO ODCZYT

Sprzątanie osieroconych zależności: `apt autoremove`

```
sudo apt autoremove
```

Usuwamy pakiety zainstalowane wyłącznie jako zależności innych, już usuniętych pakietów.

Przykład:

```
The following packages will be REMOVED:
  libgd3 libjbig0 libwebp7
After this operation, 2 155 kB disk space will be freed.
```

apt = Advanced Package Tool: menedżer pakietów Debiana i Ubuntu

TYLKO ODCZYT

Wyszukiwanie pakietu: `apt search`

```
apt search NazwaPakietu
```

Przeszukujemy nazwy i opisy pakietów w lokalnym cache metadanych, bez uprawnień administratora. Odpowiednik `dnf search`.

Przykład:

```
postgresql-16/noble 16.4-1 amd64
  object-relational SQL database, version 16 server
```

apt = Advanced Package Tool: menedżer pakietów Debiana i Ubuntu

TYLKO ODCZYT

Wyświetlenie szczegółów pakietu: `apt show`

```
apt show NazwaPakietu
```

Wersja, rozmiar, zależności i opis pakietu, dostępnego albo już zainstalowanego. Odpowiednik `dnf info`.

Przykład:

```
Package: postgresql-16
Version: 16.4-1
Depends: postgresql-client-16, postgresql-common (>= 258~)
Description: object-relational SQL database, version 16 server
```

apt = Advanced Package Tool: menedżer pakietów Debiana i Ubuntu

TYLKO ODCZYT

Wyświetlenie wszystkich zainstalowanych pakietów: `apt list --installed`

```
apt list --installed | head -3
```

Pełna lista zainstalowanych pakietów z wersją i architekturą, zwykle łączona z `grep`.

Przykład:

```
WARNING: apt does not have a stable CLI interface. Use with caution in scripts.
apt/noble,now 2.7.14build2 amd64 [installed]
bash/noble,now 5.2.21-2ubuntu4 amd64 [installed,automatic]
```

apt = Advanced Package Tool: menedżer pakietów Debiana i Ubuntu

TYLKO ODCZYT

Sprawdzenie, co czeka na aktualizację: `apt list --upgradable`

```
apt list --upgradable
```

Lista pakietów, dla których jest nowsza wersja w repozytorium niż zainstalowana.

Przykład:

```
libssl3t64/noble-updates 3.0.13-0ubuntu3.4 amd64 [upgradable from: 3.0.13-0ubuntu3.3]
```

apt = Advanced Package Tool: menedżer pakietów Debiana i Ubuntu

TYLKO ODCZYT

Przytrzymanie pakietu: `apt-mark hold`

```
sudo apt-mark hold NazwaPakietu
```

> **Pojęcie: pakiet przytrzymany (hold).** Pakiet przytrzymany to taki, który `apt upgrade` i `full-upgrade` świadomie pomijają, nawet jeśli w repozytorium czeka nowsza wersja. Blokada zostaje w bazie `dpkg`, przeżywa restart i kolejne `apt update`, dopóki ktoś jej nie zdejmie — sięgamy po to, gdy nowa wersja jądra ma znaną regresję albo chcemy zamrozić wersję do okna serwisowego.

Przykład:

```
linux-image-generic set on hold.
```

🔒 TYLKO ODCZYT

Zdjęcie przytrzymania: `apt-mark unhold`

```
sudo apt-mark unhold NazwaPakietu
```

Zdejmujemy blokadę — pakiet znów bierze udział w normalnych aktualizacjach.

Przykład:

```
Canceled hold on linux-image-generic.
```

🔒 TYLKO ODCZYT

Wyświetlenie listy przytrzymanych pakietów: `apt-mark showhold`

```
apt-mark showhold
```

Sprawdzenie, co jest zablokowane, zanim odpalimy `full-upgrade` i zdziwimy się brakiem aktualizacji.

Przykład:

```
linux-image-generic
```

🔒 TYLKO ODCZYT

Sprawdzenie, skąd pochodzi zainstalowana wersja: `apt-cache policy`

```
apt-cache policy NazwaPakietu
```

Zainstalowana wersja, wersja-kandydat i wszystkie repozytoria oferujące pakiet, z ich priorytetem — przydaje się, gdy kilka repozytoriów oferuje ten sam pakiet w różnych wersjach.

Przykład:

```
nginx:
  Installed: 1.24.0-2ubuntu7
  Candidate: 1.24.0-2ubuntu7.1
```

```
     1.24.0-2ubuntu7.1 500 noble-security/main amd64 Packages
```

TYLKO ODCZYT

Sprawdzenie, od czego zależy pakiet: `apt-cache depends`

```
apt-cache depends NazwaPakietu
```

Drzewo zależności pakietu, zanim jeszcze cokolwiek zainstalujemy.

Przykład:

```
nginx
  Depends: nginx-core
  Depends: nginx-common
```

TYLKO ODCZYT

Sprawdzenie, który pakiet dostarcza dany plik: `apt-file search`

```
apt-file search /usr/bin/envsubst
```

`apt` sam nie wie, który pakiet zawiera dany plik — trzeba osobnego pakietu `apt-file` i jego bazy (`sudo apt-file update`). Odpowiednik `dnf provides`.

Przykład:

```
gettext-base: /usr/bin/envsubst
```

TYLKO ODCZYT

Wyświetlenie listy pakietów wprost z bazy dpkg: `dpkg -l`

```
dpkg -l | grep NazwaPakietu
```

Pytamy bezpośrednio bazę `dpkg`. Pierwsza kolumna to stan: `ii` w pełni zainstalowany, `rc` usunięty z pozostawioną konfiguracją (po `remove`, nie `purge`).

Przykład:

```
ii  nginx           1.24.0-2ubuntu7   amd64   web/proxy server
rc  nginx-common    1.22.1-1ubuntu1   all     web/proxy server - common files
```

dpkg = Debian package: niskopoziomowe narzędzie instalujące paczki .deb w Debianie i Ubuntu

TYLKO ODCZYT

Wyświetlenie listy plików zainstalowanych przez pakiet: `dpkg -L`

```
dpkg -L NazwaPakietu | head -3
```

Każda ścieżka, którą pakiet zostawił na dysku. Odpowiednik `rpm -ql`.

Przykład:

```
/usr/sbin/nginx
/lib/systemd/system/nginx.service
```

dpkg = Debian package: niskopoziomowe narzędzie instalujące paczki .deb w Debianie i Ubuntu

TYLKO ODCZYT

Sprawdzenie, do jakiego pakietu należy plik: `dpkg -S`

```
dpkg -S /usr/sbin/nginx
```

Odwrotność powyższego — ścieżka na wejściu, nazwa pakietu na wyjściu. Odpowiednik `rpm -qf`.

Przykład:

```
nginx-core: /usr/sbin/nginx
```

dpkg = Debian package: niskopoziomowe narzędzie instalujące paczki .deb w Debianie i Ubuntu

TYLKO ODCZYT

Instalacja z lokalnego pliku `.deb`: `dpkg -i`

```
sudo dpkg -i ./aplikacja_1.2.3_amd64.deb
```

Instalujemy pojedynczy plik pakietu, ale **nie ściągamy** przy tym zależności z sieci — różnica względem `apt install ./plik.deb`. Jeśli zależności brakuje, instalacja kończy się błędem.

Przykład:

```
dpkg: dependency problems prevent configuration of aplikacja:
 aplikacja depends on libfoo2 (>= 2.0); however:
  Package libfoo2 is not installed.
```

dpkg = Debian package: niskopoziomowe narzędzie instalujące paczki .deb w Debianie i Ubuntu

TYLKO ODCZYT

Dokończenie przerwanej instalacji: `dpkg --configure -a`

```
sudo dpkg --configure -a
```

Kończymy nią konfigurację pakietów rozpakowanych, ale nie doprowadzonych do etapu `postinst` — typowy skutek przerwanego `dpkg -i` albo utraty zasilania. Pierwsza komenda przy komunikacie o przerwanej bazie.

Przykład:

```
Setting up aplikacja (1.2.3) ...
```

dpkg = Debian package: niskopoziomowe narzędzie instalujące paczki .deb w Debianie i Ubuntu

TYLKO ODCZYT

Weryfikacja plików zainstalowanego pakietu: `dpkg -V`

```
dpkg -V NazwaPakietu
```

Sprawdzamy sumy kontrolne plików względem tego, co zapisano przy instalacji. Cisza znaczy „zgodne”; `??5??????` znaczy zmianę pliku konfiguracyjnego albo nadpisanie binarki. Odpowiednik `rpm -V`.

Przykład:

```
??5??????   c /etc/nginx/nginx.conf
```

dpkg = Debian package: niskopoziomowe narzędzie instalujące paczki .deb w Debianie i Ubuntu

◫ ZMIENIA SYSTEM

Dodanie repozytorium PPA albo zewnętrznego: `add-apt-repository`

```
sudo add-apt-repository ppa:example/repository
```

Na Ubuntu dodajemy repozytorium PPA razem z kluczem GPG; komenda od razu odświeża listę pakietów (wymaga `software-properties-common`). Debian nie ma PPA — tam repozytorium dodaje się ręcznie plikiem w `sources.list.d/`.

Przykład:

```
Adding deb entry to /etc/apt/sources.list.d/example-ubuntu-repository-noble.sources
Adding key to /etc/apt/keyrings/example-ubuntu-repository.gpg
```

◫ **GPG** = GNU Privacy Guard: program do podpisywania i szyfrowania plików oraz sprawdzania podpisów paczek

◫ TYLKO ODCZYT

`/etc/apt/sources.list.d/` — repozytoria jako osobne pliki

```
ls /etc/apt/sources.list.d/
```

Każde dodatkowe repozytorium dostaje własny plik zamiast wspólnego `/etc/apt/sources.list` — łatwiej dodać i usunąć pojedyncze repozytorium bez ryzyka dla pliku głównego.

Przykład:

```
docker.list
ubuntu.sources
```

◫ **ls** = list: wypisz zawartość katalogu

◫ TYLKO ODCZYT

Format deb822 (`.sources`) — nowy zapis repozytoriów

```
cat /etc/apt/sources.list.d/docker.sources
```

Od Ubuntu 24.04 nowe repozytoria domyślnie zapisuje się w formacie deb822 (rozszerzenie `.sources`, klucz–wartość) zamiast jednej linii `deb ...`. Oba formaty działają równolegle.

Przykład:

```
Types: deb
URIs: https://download.docker.com/linux/ubuntu
Signed-By: /etc/apt/keyrings/docker.asc
```

cat = concatenate: sklej i wypisz pliki

TYLKO ODCZYT

`apt-key` — wycofane, i `/etc/apt/keyrings/` jako następca

```
apt-key list
```

`apt-key` zarządzał kluczami GPG we wspólnym, systemowym pęku zaufanych kluczy i właśnie dlatego został wycofany (od Debiana 11/Ubuntu 22.04): każdy klucz był automatycznie zaufany dla *wszystkich* repozytoriów, więc jedno błędnie skonfigurowane repozytorium mogło podpisać pakiet cudzym kluczem. Następca to `/etc/apt/keyrings/` — każdy klucz jako osobny plik, wskazywany wprost przez `Signed-By:` / `[signed-by=...]` danego repozytorium.

Przykład:

```
Warning: apt-key is deprecated. Manage keyring files in trusted.gpg.d instead.
```

GPG = GNU Privacy Guard: program do podpisywania i szyfrowania plików oraz sprawdzania podpisów paczek

TYLKO ODCZYT

`unattended-upgrades` — automatyczne aktualizacje bezpieczeństwa

```
sudo dpkg-reconfigure unattended-upgrades
```

Usługa instalująca w tle aktualizacje bezpieczeństwa bez udziału administratora. Konfiguracja: `/etc/apt/apt.conf.d/50unattended-upgrades` (co wolno aktualizować) i `20auto-upgrades` (czy mechanizm jest włączony).

Przykład:

```
Created symlink /etc/systemd/system/timers.target.wants/apt-daily-upgrade.timer → ...
```

◫◫ TYLKO ODCZYT

`/var/log/apt/history.log` — historia operacji pakietowych

```
tail -10 /var/log/apt/history.log
```

Każda instalacja, aktualizacja i usunięcie z datą i listą pakietów — pierwsze miejsce, żeby ustalić, co i kiedy zmieniło się w systemie.

Przykład:

```
Start-Date: 2026-09-17  09:12:03
Commandline: apt full-upgrade
Upgrade: openssl:amd64 (3.0.13-0ubuntu3.3, 3.0.13-0ubuntu3.4)
End-Date: 2026-09-17  09:14:51
```

Uważaj

- `apt upgrade` i `apt full-upgrade` **nie** są zamiennikami wywołania obu naraz — `full-upgrade` obejmuje wszystko, co zrobiłby `upgrade`, i dokłada usuwanie/podmianę pakietów.
- `apt-mark hold` przeżywa restart i kolejne `apt update` — jeśli aktualizacja „dziwnie” nic nie robi na jednym pakiecie, sprawdź `apt-mark showhold`, zanim zaczniesz szukać dalej.
- `dpkg -i` nie ściąga zależności z sieci; do tego zawsze `apt install ./plik.deb`.

◫◫ **tail** = pokaż koniec pliku, z flagą -f śledzi plik na żywo

Z życia wzięte

◫◫ Z ŻYCIA WZIĘTE

Czwartek. Nowy serwer Ubuntu w laboratorium Oli ma przerwaną instalację po awarii zasilania, a apt odmawia jakiejkolwiek dalszej pracy. Tomek uruchamia dpkg --configure -a, żeby dokończyć to, co utknęło w połowie, zanim w ogóle spróbuje czegokolwiek zainstalować

> przez apt.

Po co to istnieje

> **Pojęcie: pakiet `.deb` kontra `.rpm`.** Oba formaty robią to samo — pakują binarkę, metadane (nazwa, wersja, zależności, sumy kontrolne plików) i skrypty instalacyjne (`preinst` / `postinst` w `.deb`, `%pre` / `%post` w `.rpm`) w jeden archiwalny plik. `.deb` to archiwum `ar` zawierające dwa `tar`-y: jeden z metadanymi kontrolnymi, drugi z zawartością pakietu; `.rpm` ma własny binarny nagłówek zamiast archiwum `ar`, ale niesie te same informacje. Żaden z formatów nie jest „lepszy” — różnią się rodowodem (`.deb` od Debiana z 1993, `.rpm` od Red Hata z 1997) i narzędziami, nie możliwościami.

Debian rozwiązał ten sam problem co RHEL, tylko wcześniej: sama instalacja pakietu `.deb`/`.rpm` nic nie wie o internecie ani o brakujących zależnościach — to zadanie menedżera wysokiego poziomu. Na Debianie jest nim `apt` (Advanced Package Tool), wydany w 1998 właśnie po to, żeby zdjąć z administratora ręczne śledzenie łańcuchów zależności. Ten sam motyw dał kilka lat później `yum`, a potem `dnf` po stronie RPM-a.

Analogia

`dpkg` to **magazynier z papierowym rejestrem**, `apt` to **kierownik zaopatrzenia**.

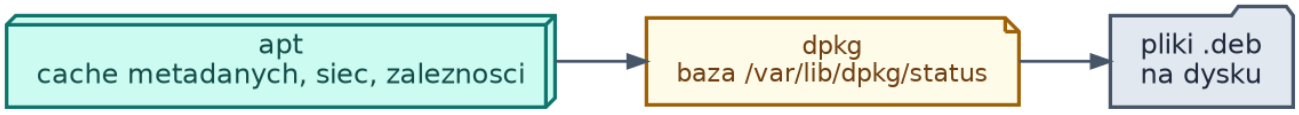
Magazynier umie dokładnie jedno: przyjąć konkretną skrzynkę, zapisać w rejestrze co w niej jest i gdzie ją postawił, albo wydać skrzynkę z powrotem i wykreślić wpis. Jeśli podasz mu skrzynkę, do której brakuje śrubek z innej dostawy — odmówi przyjęcia i powie dokładnie, czego brakuje. Nie zadzwoni sam do dostawcy.

Kierownik zaopatrzenia zna cały plan zależności między dostawami i ma dostęp do katalogu dostawców (repozytoriów). Mówisz mu „potrzebuję zamontować to urządzenie”, a on sam ustala, jakich brakujących skrzynek to wymaga, zamawia je i dopiero na końcu każe magazynierowi (`dpkg`) fizycznie przyjąć każdą skrzynkę na stan — nigdy nie dotyka rejestru sam. Stąd wynika reguła, o którą potykają się wszyscy nowi: jeśli magazynier ma zablokowany rejestr (bo poprzednia dostawa została przerwana w pół rozładunku), kierownik też nie ruszy z miejsca, dopóki magazynier nie dokończy tamtej sprawy — stąd `dpkg --configure -a` jako pierwszy krok naprawczy, zanim `apt` w ogóle spróbuje czegokolwiek.

Jak to działa

>

> **Pojęcie: baza `dpkg` kontra warstwa `apt`.** Prawda o tym, co jest zainstalowane w systemie, leży w jednym miejscu: pliku `/var/lib/dpkg/status` (plus katalog `/var/lib/dpkg/info/` z listami plików i skryptami każdego pakietu). To jest cała „baza danych” — zwykły plik tekstowy w formacie RFC822, bez silnika bazodanowego. `dpkg` czyta go i zapisuje bezpośrednio; każda operacja na pojedynczym pakiecie — rozpakowanie archiwum, uruchomienie skryptu `postinst`, aktualizacja wpisu — to `dpkg`, i tylko `dpkg`. `apt` nigdy nie modyfikuje tego pliku sam — zawsze woła `dpkg` w tle i pozwala jemu wykonać właściwą pracę.



Apt liczy zależności i sieć, a faktyczną instalację plików wykonuje zawsze dpkg. (kliknięcie otwiera pełny rozmiar)

To, co dokłada `apt`, to warstwa nad tym: własny cache metadanych repozytoriów w `/var/lib/apt/lists/` (pobierany przez `apt update`), silnik rozwiązywania zależności, który liczy, jakie pakiety trzeba doinstalować/usunąć/podbić, żeby żądanie się spięło, oraz komunikację z siecią po archiwa `.deb` do `/var/cache/apt/archives/`. Kiedy `apt install` kończy liczenie planu, przekazuje listę plików `.deb` do `dpkg`, który wykonuje faktyczną instalację jeden po drugim.

Dlatego `apt-get`/`apt` bez `dpkg` nie dzieje się nic — a `dpkg` bez `apt` działa, tylko boleśnie ręcznie: sam musisz wiedzieć, w jakiej kolejności instalować pakiety i skąd wziąć te, których brakuje.

Krok po kroku

1. Standardowy cykl aktualizacji.

```
sudo apt update
sudo apt upgrade
sudo apt autoremove
```

Odświeżenie metadanych, bezpieczna aktualizacja bez usuwania niczego, sprzątanie osieroconych zależności. To jest odruch, który powtarzasz najczęściej — analogiczny do `dnf check-update && dnf upgrade && dnf autoremove` na RHEL-u.

2. `remove` kontra `purge` — co zostaje na dysku.

```
sudo apt remove nginx
dpkg -l | grep nginx
```

```
rc  nginx-common   1.22.1-1ubuntu1   all   small, powerful, scalable web/proxy server - common
files
```

Litery `rc` w pierwszej kolumnie `dpkg -l` znaczą: pakiet **r**emoved, ale konfiguracja (**c**onfig) została. Jeśli planujesz zainstalować pakiet ponownie i chcesz stare ustawienia z powrotem — to jest dokładnie ten stan. Jeśli chcesz zniknięcia całkowitego:

```
sudo apt purge nginx
```

3. `apt` kontra `apt-get` — który do czego.

```
apt upgrade      # interaktywnie, z paskiem postępu i kolorem
apt-get upgrade  # w skrypcie/cronie — stabilne, przewidywalne wyjście
```

`apt` (bez `-get`) to nowszy, przyjazny dla człowieka interfejs — kolorowe podsumowania, pasek postępu, mniej pisania (`apt list` zamiast `dpkg -l` plus `apt-cache policy`). Ale jego własna strona podręcznika i komunikat, który wypisuje przy nieinteraktywnym użyciu, mówią wprost:

```
WARNING: apt does not have a stable CLI interface. Use with caution in scripts.
```

Format wyjścia `apt` może się zmienić między wersjami bez ostrzeżenia, bo jego zadaniem jest wygoda terminala, nie stabilność dla parserów. `apt-get` i `apt-cache` mają gwarantowany, stabilny format od dekad — dlatego skrypty, cron i automatyzacja (Ansible, Puppet, własne skrypty patchujące) powinny używać `apt-get`, `apt-cache` i `dpkg`, nigdy gołego `apt`.

Częste błędy

Skrypt w cronie napisany z `apt` zamiast `apt-get`.

```
apt-get upgrade -y >> /var/log/patch.log 2>&1
```

Skrypt działał miesiącami, aż nowa wersja `apt` zmieniła dokładnie sformatowanie jednej linii podsumowania, a parser, który jej szukał w logu, przestał działać. Naprawa jest w samym pytaniu: skrypty i automatyzacja zawsze `apt-get`/`apt-cache`, interaktywna sesja terminala — `apt`.

`apt remove` zamiast `apt purge`, potem zdziwienie starą konfiguracją.

```
$ sudo apt remove nginx
$ sudo apt install nginx
```

Po ponownej instalacji `nginx` odpala się z konfiguracją sprzed tygodnia, bo `remove` zostawił `/etc/nginx` nietknięty, a instalator zobaczył istniejące pliki konfiguracyjne i ich nie nadpisał. Sprawdzenie i naprawa:

```
dpkg -l | grep '^rc'
sudo apt purge nginx
sudo apt install nginx
```

Przerwana instalacja blokuje wszystko inne.

```
E: dpkg was interrupted, you must manually run 'sudo dpkg --configure -a' to correct the
problem.
```

Zasilanie padło albo ktoś zabił proces w połowie `apt install`. Baza `dpkg` zostaje w stanie „w trakcie” dla tego jednego pakietu, a każda kolejna komenda `apt` odmawia działania, dopóki ten stan nie zostanie domknięty:

```
sudo dpkg --configure -a
sudo apt install -f
```

Dodanie repozytorium bez klucza w nowym formacie.

```
W: GPG error: https://repo.zielona.example/apt noble InRelease: The following signatures
couldn't be verified because the public key is not available: NO_PUBKEY A1B2C3D4E5F60789
E: The repository 'https://repo.zielona.example/apt noble InRelease' is not signed.
```

Repozytorium dodane ręcznie do `/etc/apt/sources.list.d/` bez importu klucza GPG do `/etc/apt/keyrings/` i bez `Signed-By:` / `[signed-by=...]` we wpisie. `apt-key add` już nie jest właściwą odpowiedzią (patrz Fiszka) — klucz trzeba pobrać i wskazać jawnie:

```
sudo curl -fsSL https://repo.zielona.example/gpg -o /etc/apt/keyrings/example.asc
echo "deb [signed-by=/etc/apt/keyrings/example.asc] https://repo.zielona.example/apt noble
main" | sudo tee /etc/apt/sources.list.d/example.list
```

□□ Ćwiczenia

>

> Ćwiczenia poniżej wymagają maszyny z Debianem albo Ubuntu i uprawnień administratora (`sudo`) — apt jest z natury komendą do zarządzania całym systemem, nie da się jej bezpiecznie odizolować w katalogu `/tmp`.

Ćwiczenie 1 — trzy sposoby na te same informacje

```
apt show bash
apt-cache policy bash
dpkg -s bash
```

Które z trzech komend pokazuje wersję-kandydata do instalacji (czyli tę, którą dostaniesz przy najbliższym `apt upgrade`), a które tylko wersję już zainstalowaną?

Odpowiedź

`apt-cache policy bash` jako jedyna pokazuje obie wartości naraz — `Installed:` i `Candidate:` — plus pełną listę repozytoriów, z których pakiet mógłby pochodzić, z priorytetami. `apt show` i `dpkg -s` pokazują tylko metadane wybranej (zwykle zainstalowanej) wersji, bez porównania z tym, co czeka w repozytorium.

Ćwiczenie 2 — symulacja przytrzymania pakietu przed oknem serwisowym

```
sudo apt install -y sl
apt-mark showhold
sudo apt-mark hold sl
apt-mark showhold
sudo apt-get upgrade -y --dry-run | grep -i sl || echo "sl pominięty przez upgrade"
sudo apt-mark unhold sl
sudo apt purge -y sl
```

Dlaczego do sprawdzenia efektu blokady użyto `--dry-run`, a nie prawdziwego `apt upgrade`?

Odpowiedź

`--dry-run` (albo `-s`/`--simulate`) pokazuje dokładnie to, co `apt`/`apt-get` by zrobił, bez faktycznego pobierania i instalowania czegokolwiek — bezpieczny sposób na sprawdzenie działania blokady bez ryzyka rzeczywistej aktualizacji pakietów, których nie planowałeś ruszać w tym ćwiczeniu.

Pytania kontrolne

1. Czym różni się `apt remove` od `apt purge`?

Odpowiedź

`remove` usuwa binarki i pliki programu, ale zostawia pliki konfiguracyjne w `/etc` (stan `rc` w `dpkg -l`). `purge` usuwa dodatkowo te pliki konfiguracyjne. Wybór zależy od tego, czy planujesz ponowną instalację tego samego pakietu z zachowaniem ustawień.

2. Dlaczego `apt upgrade` czasem zostawia pakiet niezaktualizowany, mimo że nowa wersja jest dostępna w repozytorium?

Odpowiedź

`apt upgrade` nigdy nie usuwa ani nie instaluje nowych pakietów — jeśli aktualizacja wymagałaby jednego z tych ruchów (typowo przy jądrze), pakiet trafia na listę „kept back”, a rozwiązanie jest w `apt full-upgrade`, które to ograniczenie zdejmuje.

3. Dlaczego w skryptach i cronie powinno się używać `apt-get`, a nie `apt`?

Odpowiedź

`apt` sam ostrzega w swojej dokumentacji, że nie ma stabilnego interfejsu wiersza poleceń — jego wyjście jest zoptymalizowane pod interaktywny terminal i może się zmienić między wersjami bez zapowiedzi. `apt-get` i `apt-cache` mają gwarantowany, stabilny format wyjścia od lat i to na nich powinna opierać się automatyzacja.

4. Co się stanie, jeśli uruchomisz `apt install` w trakcie, gdy poprzednia instalacja `dpkg` została przerwana i nie dokończona?

Odpowiedź

`apt` odmówi działania z komunikatem o przerwanej bazie `dpkg` i każe najpierw uruchomić `sudo dpkg --configure -a`, żeby dokończyć konfigurację pakietów zawieszonych w połowie instalacji. Dopiero po tym `apt` wznawia normalną pracę.

Powiązane: dnf: instalacja, aktualizacja, historia, wycofywanie zmian · zypper: odpowiedniki na SLES i openSUSE · Standardowa aktualizacja systemu produkcyjnego · systemd i systemctl — jednostki, zależności, stany

Źródła: `man 8 apt` · `man 8 apt-get` · `man 1 dpkg` · `man 5 sources.list` · Debian Wiki: SourcesList (format deb822) · Debian Wiki: DropAptKey

---

Revision #15
Created 2026-09-18 06:56:59 UTC by Tomasz Konieczny
Updated 2026-10-03 19:09:36 UTC by Tomasz Konieczny