Table of Contents

$ apt

Beginner

Package manager for Debian-based distributions

apt is the primary package management command for Debian, Ubuntu, and their derivatives. It provides a user-friendly interface for installing, removing, upgrading, and searching software packages.

apt combines the most common functions of apt-get and apt-cache with a cleaner interface, progress ...

Options & Flags

`update`	Update package lists
`upgrade`	Upgrade installed packages
`install`	Install package(s)
`remove`	Remove a package
`purge`	Remove package and config files
`search`	Search for packages
`show`	Show package details
`autoremove`	Remove unused dependencies
`list --installed`	List installed packages

Practical Examples

Example: Update and upgrade

```
$ sudo apt update && sudo apt upgrade -y
```

Updates package lists and upgrades all packages.

Example: Install package

```
$ sudo apt install nginx php-fpm postgresql -y
```

Installs multiple packages, automatically confirming.

Example: Search packages

```
$ apt search "web server"
```

Searches package names and descriptions.

Example: Show package info

```
$ apt show nginx
```

Shows version, size, dependencies, and description.

Example: Remove and clean

```
$ sudo apt remove nginx && sudo apt autoremove
```

Removes nginx and any orphaned dependencies.

Tips & Best Practices

Pro Tip: Always update first: Run sudo apt update before installing packages. Without it, apt may not find the latest versions.

Note: apt vs apt-get: apt is for interactive terminal use (progress bars, colors). apt-get is for scripts (stable output format). Both work.

Warning: autoremove regularly: sudo apt autoremove removes orphaned packages. Run it after removing packages to keep the system clean.

$ apt-get

Beginner

Low-level package handling for Debian systems

apt-get is the traditional Debian/Ubuntu package management command. It handles package installation, removal, upgrading, and dependency resolution. While apt is now preferred for interactive use, apt-get remains the standard for scripts.

apt-get has stable, parseable output that does not change...

Options & Flags

`update`	Update package lists
`upgrade`	Upgrade packages
`dist-upgrade`	Smart upgrade (handles dependencies)
`install`	Install packages
`remove`	Remove packages
`purge`	Remove with config files
`autoremove`	Remove unused dependencies
`--no-install-recommends`	Skip recommended packages

Practical Examples

Example: Install in Docker

```
$ RUN apt-get update && apt-get install -y --no-install-recommends curl wget && rm -rf /var/lib/apt/lists/*
```

Standard Dockerfile pattern: update, install, cleanup.

Example: Non-interactive install

```
$ DEBIAN_FRONTEND=noninteractive apt-get install -y tzdata
```

Installs without interactive prompts (for scripts).

Example: Install specific version

```
$ sudo apt-get install nginx=1.24.0-1
```

Installs a specific version of a package.

Example: Download only

```
$ sudo apt-get install -d nginx
```

Downloads packages without installing.

Example: Fix broken dependencies

```
$ sudo apt-get install -f
```

Attempts to fix broken package dependencies.

Tips & Best Practices

Pro Tip: Use in scripts, apt for terminal: apt-get has stable output for scripts and automation. apt has nicer output for interactive use.

Note: Dockerfile pattern: Always combine update and install in one RUN command in Dockerfiles, and clean up /var/lib/apt/lists/* to reduce image size.

Warning: -y flag in scripts: Always use -y in scripts to auto-confirm. Without it, apt-get waits for input that never comes.

$ dnf

Beginner

Package manager for RPM-based distributions (Fedora, RHEL 8+)

dnf (Dandified YUM) is the package manager for Fedora, RHEL 8+, CentOS Stream, and Rocky/AlmaLinux. It replaces yum with better performance, dependency resolution, and a cleaner API.

dnf handles package installation, removal, upgrading, and repository management. It uses RPM packages and resolve...

Options & Flags

`install`	Install packages
`remove`	Remove packages
`update`	Update packages
`search`	Search for packages
`info`	Show package details
`list installed`	List installed packages
`group install`	Install package group
`autoremove`	Remove unused dependencies

Practical Examples

Example: Install package

```
$ sudo dnf install -y nginx php postgresql-server
```

Installs packages with automatic confirmation.

Example: Update system

```
$ sudo dnf update -y
```

Updates all packages to latest versions.

Example: Search packages

```
$ dnf search "web server"
```

Searches package names and summaries.

Example: Package info

```
$ dnf info nginx
```

Shows version, repository, size, and description.

Example: Install dev tools

```
$ sudo dnf group install "Development Tools"
```

Installs compiler, make, and other development utilities.

Tips & Best Practices

Pro Tip: dnf replaces yum: dnf is the successor to yum. On RHEL 8+, yum is actually an alias for dnf. Use dnf for all new work.

Note: Equivalent to apt: dnf install = apt install. dnf update = apt upgrade. dnf search = apt search. Same concepts, different ecosystem.

Warning: EPEL for extra packages: Many packages require EPEL repository: sudo dnf install epel-release. This adds thousands of additional packages.

$ dpkg

Intermediate

Debian package manager for .deb files

dpkg is the low-level Debian package manager that installs, removes, and inspects .deb package files. While apt handles dependencies automatically, dpkg works directly with individual .deb files.

dpkg is used when you download a .deb file directly (from a website or custom build) and need to ins...

Options & Flags

`-i`	Install a .deb package
`-r`	Remove a package
`-P`	Purge (remove with config)
`-l`	List installed packages
`-L`	List files installed by package
`-S`	Find which package owns a file
`-s`	Show package status
`--configure -a`	Fix interrupted installations

Practical Examples

Example: Install .deb file

```
$ sudo dpkg -i google-chrome-stable.deb
```

Installs a downloaded .deb package.

Example: Fix dependencies

```
$ sudo dpkg -i package.deb; sudo apt install -f
```

Installs a .deb, then resolves any missing dependencies.

Example: List installed packages

```
$ dpkg -l | grep php
ii  php8.2  8.2.0-1  amd64  PHP scripting language
```

Lists all installed packages matching "php".

Example: Find package for file

```
$ dpkg -S /etc/nginx/nginx.conf
nginx-common: /etc/nginx/nginx.conf
```

Shows which package installed a specific file.

Example: List package files

```
$ dpkg -L nginx
```

Shows all files installed by the nginx package.

Tips & Best Practices

Pro Tip: Install then fix deps: sudo dpkg -i file.deb && sudo apt install -f. The apt install -f resolves missing dependencies automatically.

Note: dpkg -S for file ownership: dpkg -S /path/to/file tells you which package installed that file - very useful for troubleshooting.

Warning: No dependency resolution: dpkg does not install dependencies. Use apt install package to get dependencies resolved automatically.

$ flatpak

Beginner

Install and manage Flatpak applications

flatpak is a universal Linux package manager for desktop applications. Like snap, it provides sandboxed, cross-distribution packages, but with a focus on desktop apps and a decentralized repository model.

flatpak applications run in sandboxed environments with configurable permissions. The prima...

Options & Flags

`install`	Install an application
`uninstall`	Remove an application
`list`	List installed apps
`update`	Update all apps
`search`	Search for apps
`run`	Run an application
`remote-add`	Add a repository

Practical Examples

Example: Add Flathub

```
$ flatpak remote-add --if-not-exists flathub https://flathub.org/repo/flathub.flatpakrepo
```

Adds the main Flathub repository.

Example: Install app

```
$ flatpak install flathub org.gimp.GIMP
```

Installs GIMP from Flathub.

Example: List installed

```
$ flatpak list --app
```

Shows all installed flatpak applications.

Example: Update all

```
$ flatpak update -y
```

Updates all installed flatpak applications.

Example: Search apps

```
$ flatpak search firefox
```

Searches Flathub for Firefox.

Tips & Best Practices

Pro Tip: Flathub is the main repo: Add Flathub first - it has the largest collection of flatpak applications.

Note: flatpak vs snap: flatpak focuses on desktop apps with decentralized repos. snap supports server apps and has centralized store (Canonical).

Warning: Sandbox limitations: Flatpak sandbox may limit file access. Use flatseal to manage application permissions graphically.

$ pip

Beginner

Install Python packages from PyPI

pip is the package installer for Python. It downloads and installs Python packages from the Python Package Index (PyPI) and other repositories. pip manages Python dependencies for applications and development.

pip handles package installation, upgrades, removal, and dependency resolution. It can...

Options & Flags

`install`	Install packages
`install -r`	Install from requirements file
`uninstall`	Remove a package
`list`	List installed packages
`freeze`	Output installed packages in requirements format
`show`	Show package details
`install --upgrade`	Upgrade a package
`install --user`	Install for current user only

Practical Examples

Example: Install package

```
$ pip install flask
```

Installs Flask from PyPI.

Example: Install from requirements

```
$ pip install -r requirements.txt
```

Installs all packages listed in requirements.txt.

Example: Create requirements file

```
$ pip freeze > requirements.txt
```

Exports current packages and versions.

Example: Upgrade pip

```
$ pip install --upgrade pip
```

Updates pip itself to the latest version.

Example: Virtual environment workflow

```
$ python -m venv venv && source venv/bin/activate && pip install -r requirements.txt
```

Creates a virtual environment and installs dependencies.

Tips & Best Practices

Warning: Always use virtual environments: Never pip install globally. Use python -m venv venv to create isolated environments per project.

Pro Tip: pip freeze for reproducibility: pip freeze > requirements.txt captures exact versions. Share this file so others get the same packages.

Note: pip3 vs pip: On systems with both Python 2 and 3, use pip3 to ensure Python 3. Or python3 -m pip for certainty.

$ rpm

Intermediate

RPM Package Manager for .rpm files

rpm (Red Hat Package Manager) is the low-level package tool for RHEL, CentOS, Fedora, SUSE, and other RPM-based distributions. It installs, queries, verifies, and removes .rpm packages.

rpm works directly with .rpm files and does not resolve dependencies automatically (use dnf/yum for that). It ...

Options & Flags

`-i`	Install a package
`-U`	Upgrade (or install) a package
`-e`	Erase (remove) a package
`-q`	Query installed package
`-qa`	Query all installed packages
`-ql`	List files in installed package
`-qf`	Find package owning a file
`-V`	Verify package integrity

Practical Examples

Example: Install RPM

```
$ sudo rpm -ivh package.rpm
```

Installs with verbose output and progress bar.

Example: Query all packages

```
$ rpm -qa | grep nginx
nginx-1.24.0-1.el8.x86_64
```

Lists all installed packages matching "nginx".

Example: List package files

```
$ rpm -ql nginx
/etc/nginx/nginx.conf\n/usr/sbin/nginx
```

Shows all files installed by nginx.

Example: Find file owner

```
$ rpm -qf /etc/nginx/nginx.conf
nginx-1.24.0-1.el8.x86_64
```

Shows which package installed the file.

Example: Verify package

```
$ rpm -V nginx
```

Checks if package files have been modified.

Tips & Best Practices

Pro Tip: Use -Uvh for install/upgrade: rpm -Uvh works for both new installs and upgrades. Prefer it over -ivh which fails if the package is already installed.

Note: rpm -qf for file ownership: rpm -qf /path/to/file tells you which package installed it - the RPM equivalent of dpkg -S.

Warning: No dependency resolution: rpm does not install dependencies. Use dnf install package.rpm to get automatic dependency resolution.

$ snap

Beginner

Install and manage snap packages

snap installs and manages snap packages - self-contained application bundles that include all dependencies. Snaps update automatically, run in sandboxes, and work across many Linux distributions.

Snaps are designed for cross-distribution compatibility. A single snap works on Ubuntu, Fedora, Debi...

Options & Flags

`install`	Install a snap
`remove`	Remove a snap
`list`	List installed snaps
`find`	Search for snaps
`refresh`	Update snaps
`info`	Show snap details
`--classic`	Install with classic confinement (no sandbox)

Practical Examples

Example: Install snap

```
$ sudo snap install code --classic
```

Installs VS Code with classic confinement (full filesystem access).

Example: List installed

```
$ snap list
Name      Version   Rev    Tracking\nfirefox   120.0     3432   latest/stable
```

Shows all installed snaps with versions.

Example: Search snaps

```
$ snap find node
```

Searches the Snap Store for Node.js related snaps.

Example: Update all snaps

```
$ sudo snap refresh
```

Updates all snaps to latest versions.

Example: Remove snap

```
$ sudo snap remove firefox
```

Removes the snap and its data.

Tips & Best Practices

Note: Classic vs strict: Strict snaps run in a sandbox (limited filesystem access). Classic snaps (--classic) have full access. IDEs and dev tools often need --classic.

Pro Tip: Auto-updates: Snaps update automatically. Use snap refresh --hold to pause updates. snap refresh --unhold to resume.

Warning: Startup can be slower: Snaps may start slower than native packages because they mount a squashfs filesystem. This is a known trade-off.

$ yum

Beginner

Package manager for older RPM-based systems (legacy)

yum (Yellowdog Updater Modified) is the legacy package manager for RHEL, CentOS, and Fedora. It manages RPM packages with automatic dependency resolution. On RHEL 8+, yum is an alias for dnf.

yum has been the standard Red Hat package manager for over a decade. While dnf is the official successor...

Options & Flags

`install`	Install packages
`remove`	Remove packages
`update`	Update packages
`search`	Search for packages
`info`	Show package info
`list installed`	List installed packages
`groupinstall`	Install a package group

Practical Examples

Example: Install package

```
$ sudo yum install -y nginx
```

Installs nginx with auto-confirmation.

Example: Update system

```
$ sudo yum update -y
```

Updates all installed packages.

Example: Search packages

```
$ yum search php
```

Searches for PHP-related packages.

Example: Clean cache

```
$ sudo yum clean all
```

Clears cached package data to free space.

Example: Install EPEL

```
$ sudo yum install epel-release
```

Installs the EPEL repository for additional packages.

Tips & Best Practices

Note: yum is dnf on RHEL 8+: On RHEL 8+, CentOS Stream, and Fedora, yum is a symlink to dnf. Your yum commands automatically use dnf.

Pro Tip: Use dnf for new work: For new scripts and documentation, use dnf instead of yum. dnf is faster and better at dependency resolution.

Warning: RHEL 7 end of life: RHEL/CentOS 7 (which uses real yum) reached end of life. Upgrade to RHEL 8/9 which use dnf.