



POP!_OS 24.04 LTS · COSMIC · APT · FLATPAK

Pop!_OS Handbook

For advanced users

Run the system from the inside. APT and dpkg in depth, PPAs, the recovery partition and kernelstub, systemd, AppArmor, ufw, NVIDIA and hybrid graphics, and recovering a system that won't boot.

CONTENTS

1. APT in depth
2. dpkg queries
3. Repositories and PPAs
4. Recovery partition and kernelstub
5. Firewall: ufw
6. AppArmor
7. systemd services
8. Logs: journalctl
9. Networking: nmcli, ip, ss
10. Users and permissions
11. SSH
12. Disks, swap and snapshots
13. Drivers: NVIDIA and hybrid graphics
14. Podman, Docker, Distrobox
15. Text processing and pipelines
16. Archives, backups, syncing
17. Customizing the shell
18. Performance and power
19. Troubleshooting and recovery

Yellow italic text (like *<package>*) marks the parts you fill in. In the terminal, paste is Ctrl+Shift+V and copy is Ctrl+Shift+C. Commands that start with **sudo** change the system; read what they do before running them.



sudogrimoire.com/popos/

Free, searchable handbooks for Linux, Windows and macOS

APT in depth

APT settings live in `/etc/apt/apt.conf.d/`, repository definitions in `/etc/apt/sources.list.d/`, pinning rules in `/etc/apt/preferences.d/`.

```
$ apt policy <package>
```

Installed version, candidate version, which repository it comes from and priorities.

```
$ apt-cache madison <package>
```

All versions available in the repositories.

```
$ sudo apt install <package>=<version>
```

Installs a specific version (also used to downgrade).

```
$ sudo apt-mark hold <package>
```

Holds a package back from upgrades. `unhold` releases it, `showhold` lists holds.

```
$ apt-mark showmanual
```

Packages you installed yourself (not pulled in as dependencies).

```
$ apt-cache depends <package>
```

A package's dependencies. Reverse: `apt-cache rdepends --installed <package>`.

```
$ sudo apt install --no-install-recommends  
<package>
```

Installs without the extra Recommends packages.

```
$ sudo apt install apt-file && sudo apt-file  
update
```

Lets you search the files of packages that aren't installed.

```
$ apt-file search <file-name>
```

Finds which package provides a file or command.

```
$ less /var/log/apt/history.log
```

APT history: what was installed or removed, and when. Older entries: `zless history.log.1.gz`.

```
$ sudo apt clean
```

Empties the downloaded package cache (`/var/cache/apt/archives`). `autoclean` removes only obsolete ones.

```
$ sudo dpkg-reconfigure <package>
```

Asks a package's setup questions again.
Examples: `tzdata`, `locales`, `keyboard-configuration`.

```
$ sudo apt install needrestart
```

Tells you which services need restarting after updates.

```
$ sudo apt install unattended-upgrades && sudo  
dpkg-reconfigure -plow unattended-upgrades
```

Installs security updates automatically. Logs are under `/var/log/unattended-upgrades/`.

```
$ apt modernize-sources
```

Converts old one-line `sources.list` files to the new deb822 (`.sources`) format (APT 3.0 and later; run with `sudo`).

dpkg queries

`dpkg` is the low-level package tool underneath APT. Install with APT; `dpkg` is ideal for queries and repairs.

```
$ dpkg -l | grep <word>
```

Lists installed packages with their state (ii = installed, rc = removed but config remains).

```
$ dpkg -L <package>
```

Files of an installed package.

```
$ dpkg -S <file-path>
```

Tells you which package a file belongs to.
Example: `dpkg -S /usr/bin/ls`.

```
$ dpkg -s <package>
```

A package's status, version and dependencies.

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

Finishes interrupted installations (the "dpkg was interrupted" error).

```
$ sudo apt --fix-broken install
```

Fixes a broken state by installing missing dependencies.

```
$ dpkg -l | awk '/^rc/{print $2}' | xargs -r  
sudo apt purge -y
```

Cleans up config files left over from removed packages.

```
$ dpkg-query -W -f='${Installed-  
Size}\t${Package}\n' | sort -n | tail -20
```

The 20 largest packages (KB).

```
$ sudo dpkg --add-architecture i386 && sudo apt  
update
```

Enables 32-bit packages (for Steam, Wine).

```
$ sudo apt install debsums && sudo debsums -s
```

Checks package files for corruption; prints only problems.

Repositories and PPAs

Pop!_OS uses its own repositories on top of Ubuntu's. PPAs work as on Ubuntu.

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

Repository files, including Pop!_OS's own.

```
$ sudo add-apt-repository ppa:<user>/<ppa>
```

Adds a PPA.

```
$ sudo add-apt-repository --remove ppa:  
<user>/<ppa>
```

Removes a PPA.

WARNING PPAs aren't reviewed by System76 or Canonical; add only PPAs you trust, built for Ubuntu 24.04 ("noble").

Recovery partition and kernelstub

Pop!_OS uses systemd-boot (not GRUB) and keeps a recovery partition with a copy of the installer.

TIP Hold **Space** while the computer starts to see the boot menu; choose **Pop!_OS Recovery** to start the installer from the disk.

TIP In recovery, **Refresh Install** reinstalls the system and keeps your home folder.

```
$ sudo kernelstub -a "<option>"
```

Adds a kernel parameter (for example `nomodeset`).

```
$ sudo kernelstub -d "<option>"
```

Removes a kernel parameter.

```
$ sudo kernelstub -p
```

Shows the current boot settings.

Firewall: ufw

Pop!_OS has no firewall installed by default. ufw (Uncomplicated Firewall) is a simple front end to nftables.

```
$ sudo apt install ufw
```

Installs ufw.

```
$ sudo ufw status verbose
```

Status and rules.

```
$ sudo ufw default deny incoming && sudo ufw  
default allow outgoing
```

Block incoming, allow outgoing (a sensible desktop default).

```
$ sudo ufw enable
```

Turns the firewall on and keeps it on at boot.

```
$ sudo ufw allow ssh
```

Allows SSH (22). `sudo ufw limit ssh` slows down brute-force attempts.

```
$ sudo ufw allow 8080/tcp
```

Opens a port.

```
$ sudo ufw allow from 192.168.1.0/24 to any port  
445
```

Allows a port only from the local network.

```
$ sudo ufw status numbered
```

Lists rules with numbers.

```
$ sudo ufw delete <number>
```

Deletes a rule by number.

```
$ sudo ufw app list
```

Ready-made app profiles defined by packages.

```
$ sudo apt install gufw
```

Graphical interface.

WARNING Docker writes its own iptables/nftables rules, so ports it publishes bypass ufw. If you use Docker, bind ports to localhost like `127.0.0.1:8080:80`.

AppArmor

Pop!_OS limits what programs can access with AppArmor profiles. If a program says “Permission denied” while file permissions look right, AppArmor may be the cause.

```
$ sudo aa-status
```

Loaded profiles and their mode (enforce / complain).

```
$ sudo journalctl -k --grep=apparmor
```

Operations AppArmor blocked (“DENIED” lines).

```
$ sudo apt install apparmor-utils
```

Installs aa-complain, aa-enforce, aa-logprof and friends.

```
$ sudo aa-complain /etc/apparmor.d/<profile>
```

Puts a profile in log-only mode (for diagnosis).

```
$ sudo aa-enforce /etc/apparmor.d/<profile>
```

Puts a profile back into enforcing mode.

```
$ sudo aa-logprof
```

Looks at the logged denials and interactively suggests rules to add.

```
$ sudo apparmor_parser -r  
/etc/apparmor.d/<profile>
```

Reloads an edited profile. Local additions go under `/etc/apparmor.d/local/`.

systemd services

systemd manages services (background daemons), timers and the boot process.

```
$ systemctl status <service>
```

Shows status, the latest log lines and the PID.

```
$ sudo systemctl start|stop|restart <service>
```

Starts / stops / restarts a service.

```
$ sudo systemctl enable --now <service>
```

Starts it at boot and starts it now.

```
$ sudo systemctl disable --now <service>
```

Stops it from starting at boot and stops it now.

```
$ sudo systemctl mask <service>
```

Prevents the service from being started at all.
`unmask` undoes it.

```
$ systemctl --failed
```

Lists failed units.

```
$ systemctl list-units --type=service --  
state=running
```

Running services.

```
$ systemctl list-unit-files --state=enabled
```

Units that start at boot.

```
$ systemctl list-timers
```

Scheduled jobs (systemd's answer to cron).

```
$ systemctl cat <service>
```

Shows the unit file and any drop-ins.

```
$ sudo systemctl edit <service>
```

Creates an override (drop-in) without touching the packaged file.

```
$ sudo systemctl daemon-reload
```

Makes systemd reread unit files after you change them.

```
$ systemctl --user enable --now <service>
```

User services; definitions live under
`~/.config/systemd/user/`.

```
$ loginctl enable-linger $USER
```

Lets user services run even when you are not logged in.

```
$ systemd-analyze blame
```

Lists how long each service took at boot.

```
$ systemd-analyze critical-chain
```

Shows the critical path of the boot.

Logs: journalctl

System logs are kept in the systemd journal. Pop!_OS keeps the systemd journal; use journalctl to read logs.

```
$ journalctl -b
```

Everything since this boot.

```
$ journalctl -b -1 -p err
```

Errors from the previous boot. The first place to look after a crash.

```
$ journalctl -u <service> -f
```

Follows a service's log live.

```
$ journalctl -p warning --since "1 hour ago"
```

Warnings and worse from the last hour.

```
$ journalctl --since today --grep "<word>"
```

Searches today's log for text.

```
$ journalctl -k
```

Kernel messages only (like dmesg).

```
$ journalctl --list-boots
```

Lists recorded boots.

```
$ journalctl --disk-usage
```

Space used by the logs.

```
$ sudo journalctl --vacuum-time=2weeks
```

Deletes logs older than two weeks. `--vacuum-size=500M` also works.

```
$ coredumpctl list
```

Records of crashed programs (if systemd-coredump is installed). `coredumpctl info <PID>` gives details.

```
$ sudo dmesg -w
```

Follows the kernel ring buffer live (to see what happens when you plug in USB).

Networking: nmcli, ip, ss

On desktop installs NetworkManager runs the network; its command-line tool is `nmcli` and its text UI is `nmtui`.

```
$ nmcli device status
```

Network devices and their state.

```
$ nmcli device wifi list
```

Visible Wi-Fi networks with signal strength.

```
$ nmcli device wifi connect "<SSID>" password "<password>"
```

Connects to a Wi-Fi network.

```
$ nmcli connection show
```

Saved connections.

```
$ nmcli connection up|down "<name>"
```

Brings a connection up / down.

```
$ sudo nmcli connection modify "<name>"  
ip4.method manual ip4.addresses  
192.168.1.50/24 ip4.gateway 192.168.1.1  
ip4.dns "1.1.1.1 9.9.9.9"
```

Sets a static IP. Then `nmcli connection up "<name>"`.

```
$ nmtui
```

Menu-driven text UI; useful without a graphical desktop.

```
$ sudo hostnamectl set-hostname <new-name>
```

Changes the computer's name.

```
$ ip -br a
```

Interfaces and IP addresses, briefly.

```
$ ip route
```

Routing table and default gateway.

```
$ resolvectl status
```

DNS servers (when systemd-resolved is used). Otherwise `cat /etc/resolv.conf`.

```
$ ss -tulpn
```

Listening TCP/UDP ports and which program owns them (sudo to see the program).

```
$ ping -c 4 <address>
```

Reachability test.

```
$ tracepath <address>
```

The route packets take.

```
$ curl -I https://<site>
```

Fetches a site's HTTP headers.

```
$ dig <domain>
```

DNS lookup (bind9-dnsutils package).

Users and permissions

On this system sudo rights come from membership in the `sudo` group.

```
$ id
```

Your user ID and groups.

```
$ sudo useradd -m -G sudo -s /bin/bash <user>
```

New user with a home folder and sudo rights.
Then `sudo passwd <user>`.

```
$ sudo usermod -aG <group> <user>
```

Adds a user to a group (without `-a` it removes them from other groups). Takes effect after logging in again.

```
$ sudo userdel -r <user>
```

Deletes a user along with their home folder.

```
$ passwd
```

Changes your own password.

```
$ chmod +x <file>
```

Makes a file executable.

```
$ chmod 644 <file>
```

Owner reads/writes, others read. 755 is typical for folders and scripts.

```
$ chmod -R u+rwX,go-rwx <folder>
```

Makes a folder accessible only to its owner.

```
$ sudo chown -R <user>:<group> <path>
```

Changes the owner.

```
$ setfacl -m u:<user>:rw <file>
```

Gives a specific user extra permissions (ACL).
`getfacl` shows them.

```
$ sudo -i
```

Opens a root shell. Type exit when done.

```
$ sudo visudo -f /etc/sudoers.d/<name>
```

Edits sudo rules with syntax checking.

TIP On Pop!_OS the root account has no password (it is locked); use `sudo -i` for a root shell.

SSH

```
$ sudo apt install openssh-server
```

Installs the SSH server; the service (ssh) starts automatically. Status: `systemctl status ssh`.

```
$ ssh-keygen -t ed25519 -C "<Label>"
```

Creates a key pair (`~/.ssh/id_ed25519`).

```
$ ssh-copy-id <user>@<server>
```

Uploads your public key to the server; after that you log in without a password.

```
$ ssh <user>@<server> -p <port>
```

Connects.

```
$ ssh -L 8080:localhost:80 <user>@<server>
```

Tunnels local port 8080 to port 80 on the server.

```
$ scp <file> <user>@<server>:~/
```

Copies a file to the server. `-r` for folders.

TIP Define short names in `~/.ssh/config` :

```
Host myserver · HostName 192.168.1.10 · User user · IdentityFile ~/.ssh/id_ed25519
```

Then just `ssh myserver`.

TIP To turn off password logins, write `PasswordAuthentication no` into `/etc/ssh/sshd_config.d/10-local.conf` and restart the SSH service.

Disks, swap and snapshots

Pop!_OS uses ext4 by default. Timeshift is the most common tool for system snapshots: it works in rsync mode on ext4, and in btrfs mode on Btrfs with Ubuntu-style `@` / `@home` subvolumes.

```
$ sudo apt install timeshift
```

Installs Timeshift; on first launch a wizard asks for the target disk and schedule.

```
$ sudo timeshift --create --comments "  
<description>"
```

Takes a snapshot manually. A good habit before a big update.

```
$ sudo timeshift --list
```

Lists snapshots.

```
$ sudo timeshift --restore
```

Interactive restore. It can also run from a live USB if the system won't boot.

```
$ lsblk -f && findmnt /
```

Disks, file systems and where root is mounted.

```
$ sudoedit /etc/fstab
```

Disks mounted at boot. `sudo blkid` gives the UUIDs. A mistake breaks booting; test with `sudo mount -a` afterwards.

```
$ swapon --show
```

Active swap areas.

```
$ sudo fallocate -l 4G /swapfile && sudo chmod  
600 /swapfile && sudo mkswap /swapfile && sudo  
swapon /swapfile
```

Creates a 4 GB swap file. To make it permanent, add `/swapfile none swap sw 0 0` to fstab.

```
$ sudo apt install systemd-zram-generator
```

Compressed swap in RAM (zram); configure it in `/etc/systemd/zram-generator.conf`.

```
$ sudo apt install smartmontools && sudo  
smartctl -a /dev/<sda>
```

Disk health report (SMART). For NVMe use `/dev/nvme0`.

```
$ sudo tune2fs -l /dev/<sda2> | grep -i -E  
'state|mount count'
```

State of an ext4 partition.

WARNING Don't run `fsck` on a mounted partition. To check the root partition, boot from a live USB, or add `fsck.mode=force` to the kernel line in GRUB for one boot.

Drivers: NVIDIA and hybrid graphics

System76 packages the NVIDIA driver and a power tool that switches graphics modes on laptops.

```
$ sudo apt install system76-driver-nvidia
```

Installs the NVIDIA driver.

```
$ system76-power graphics
```

Shows the graphics mode on hybrid laptops.

```
$ sudo system76-power graphics hybrid
```

Switches mode: `integrated`, `hybrid`, `nvidia` or `compute`. Restart afterwards.

```
$ nvidia-smi
```

GPU status and usage.

WARNING Don't install the NVIDIA driver from the `.run` file on [nvidia.com](https://www.nvidia.com); it conflicts with the package manager.

Podman, Docker, Distrobox

Distrobox lets you use other distributions' tools without cluttering the host; Podman or Docker run services. Podman is largely compatible with Docker commands and doesn't need root.

```
$ sudo apt install podman distrobox
```

Installs Podman and Distrobox.

```
$ distrobox create -i  
registry.fedoraproject.org/fedora:latest -n  
<name>
```

Creates a container of another distribution (e.g. `-i archlinux`, `-i ubuntu:24.04`).

```
$ distrobox enter <name>
```

Enters it; your home folder is shared. Inside, `distrobox-export --app <app>` adds the app to the host menu.

```
$ podman run -it --rm debian bash
```

Starts a throwaway container.

```
$ podman ps -a
```

All containers.

```
$ podman images
```

Local images. `podman system prune -a` removes unused ones.

```
$ podman run -d -p 8080:80 -v  
~/site:/usr/share/nginx/html  
docker.io/library/nginx
```

Runs a service with a mounted folder.

```
$ sudo apt install docker.io docker-compose-v2
```

Installs Docker from the Ubuntu repositories Pop!_OS uses. For Docker's own repository, follow its Ubuntu "noble" instructions.

```
$ sudo usermod -aG docker $USER
```

Use Docker without sudo (log in again). This group is equivalent to root access.

TIP To run a Podman container as a systemd service, use Quadlet: write a `.container` file under `~/.config/containers/systemd/`, run `systemctl --user daemon-reload` and start it.

Text processing and pipelines

| passes one command's output to the next. > writes to a file (overwriting), >> appends, 2>&1 sends errors to the same place. && runs if the previous command succeeded, || if it failed.

```
$ grep -rni "<text>" <folder>
```

Recursive, case-insensitive search with line numbers.

```
$ grep -v "^#" <file> | grep -v "^$"
```

Drops comments and blank lines; handy for reading config files.

```
$ sed -i 's/<old>/<new>/g' <file>
```

Find and replace in a file. Try without -i first and check the output.

```
$ awk '{print $1, $3}' <file>
```

Prints the 1st and 3rd column of each line.

```
$ sort <file> | uniq -c | sort -rn | head
```

Most frequently repeated lines.

```
$ wc -l <file>
```

Line count.

```
$ cut -d: -f1 /etc/passwd
```

First colon-separated field (user names).

```
$ find . -name "*.log" -mtime +7 -delete
```

Deletes .log files older than 7 days.

```
$ find . -type f -name "*.jpg" -print0 | xargs  
-0 -I{} cp {} ~/Pictures/
```

Copies found files one by one (safe with spaces in names).

```
$ <command> | tee <file>
```

Writes output to both the screen and a file.

```
$ echo "<line>" | sudo tee -a /etc/<file>
```

Appends to a root-owned file (sudo echo >> doesn't work).

```
$ diff -u <old> <new>
```

Differences between two files.

```
$ watch -n 2 <command>
```

Reruns a command every 2 seconds.

```
$ curl -s <URL> | jq '.'
```

Pretty-prints JSON (jq package).

Archives, backups, syncing

```
$ tar -czvf <archive>.tar.gz <folder>
```

Archives a folder with gzip.

```
$ tar --zstd -cvf <archive>.tar.zst <folder>
```

Archives with zstd; faster than gzip and compresses better.

```
$ tar -xvf <archive>
```

Extracts any tar archive. Use `-C <folder>` for a target folder.

```
$ tar -tvf <archive>
```

Lists the contents without extracting.

```
$ zip -r <archive>.zip <folder>
```

Creates a zip. Extract with `unzip <archive>.zip`.

```
$ 7z x <archive>.7z
```

Extracts 7z / rar and similar formats (7zip package).

```
$ rsync -avh --progress <source>/ <target>/
```

Syncs a folder; copies only what changed. The trailing / means "the contents".

```
$ rsync -avh --delete --dry-run <source>/  
<target>/
```

Mirror, deleting from the target what's not in the source; see what would happen first with `--dry-run`.

```
$ rsync -avz -e ssh <source>/ <user>@<server>:  
<path>/
```

Remote backup over SSH.

```
$ rclone config
```

Sets up cloud targets such as Google Drive, OneDrive and S3; sync with `rclone sync`.

Customizing the shell

Bash settings live in `~/.bashrc` in your home folder. Add aliases and variables at the end of the file.

```
$ nano ~/.bashrc
```

Opens the shell configuration file.

```
alias update='sudo apt update && sudo apt full-  
upgrade && flatpak update'
```

System update in one word. Write it into the file and save.

```
alias ll='ls -lah --group-directories-first'
```

A handy detailed listing.

```
$ source ~/.bashrc
```

Loads the changes without opening a new terminal.

```
export PATH="$HOME/.local/bin:$PATH"
```

Adds your own scripts folder to PATH.

```
$ echo $SHELL
```

The shell you are using.

```
$ sudo apt install zsh && chsh -s /usr/bin/zsh
```

Installs Zsh and makes it the default (log in again). Same method for Fish.

```
$ type <command>
```

Shows whether a name is an alias, a function or a file.

TIP When writing a script, put `#!/usr/bin/env bash` on the first line and `set -euo pipefail` on the second; it stops on errors and catches undefined variables. Then `chmod +x script.sh`.

Performance and power

Pop!_OS manages power profiles with `system76-power`; the battery menu in the top panel shows them.

```
$ powerprofilesctl list
```

Power profiles and which one is active (`power-profiles-daemon`).

```
$ powerprofilesctl set performance
```

Switches profile: `power-saver` / `balanced` / `performance`.

```
$ sudo apt install lm-sensors
```

CPU and disk temperatures; run `sensors` after installing.

```
$ sudo apt install btop nvidia-smi
```

Advanced system and GPU monitors.

```
$ sudo iotop -o
```

Processes writing the most to disk (`iotop` package).

```
$ systemd-analyze
```

Total boot time (firmware, loader, kernel, userspace).

```
$ sudo fstrim -av
```

Runs TRIM on SSDs. To do it weekly: `sudo systemctl enable --now fstrim.timer`.

```
$ upower -i $(upower -e | grep BAT)
```

Battery health, capacity and charge information.

```
$ free -h && swapon --show
```

RAM and swap areas.

Troubleshooting and recovery

What to try, in order, when the system won't boot or an update broke something.

TIP 1. Boot an older kernel. In GRUB, pick the previous kernel under “Advanced options for Pop!_OS”. If the menu doesn't appear, press Esc (UEFI) or Shift (BIOS) while booting.

TIP 2. Switch to a text console. Ctrl+Alt+F3 (F2–F6). Log in and check errors with `journalctl -b -p err`.

```
$ sudo systemctl restart display-manager
```

Restarts the graphical login manager (GDM, SDDM, LightDM); the open session closes.

```
$ sudo dpkg --configure -a && sudo apt --fix-broken install
```

Finishes interrupted updates.

TIP 3. Recovery mode. GRUB > Advanced options > the “(recovery mode)” entry. Choose “root” from the menu to get a shell; if root is mounted read-only, run `mount -o remount,rw /`.

TIP 4. chroot from a live USB. Boot a live USB and run the following in a terminal, in order. Adjust disk names to your system using `lsblk`.

```
$ sudo mount /dev/<sda2> /mnt && sudo mount /dev/<sda1> /mnt/boot/efi
```

Mounts the root and EFI partitions (add `-o subvol=@` for Btrfs).

```
$ for d in dev proc sys run; do sudo mount --rbind /$d /mnt/$d; done && sudo chroot /mnt
```

Switches into the installed system; apt, update-grub and update-initramfs work inside.

```
$ grub-install && update-grub
```

Reinstalls the boot loader inside the chroot (on UEFI systems).

```
$ sudo timeshift --restore
```

If Timeshift is installed, you can return to the last good snapshot from a live USB.

Commands are written for Pop!_OS 24.04 LTS (based on Ubuntu 24.04 LTS; COSMIC, APT, Flatpak). Menu names follow the English interface. Commands that start with `sudo` change the system; read what they do before running them.