FEDORA WORKSTATION · GNOME · DNF5 · BTRFS

Fedora Handbook

For advanced users

Run the system from the inside. DNF5 history, systemd, SELinux, firewalld, Btrfs snapshots, NVIDIA drivers, containers and recovering a system that won't boot. Continue here once you've finished the beginner section.

CONTENTS

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.

DNF5 in depth

Since Fedora 41 the `dnf` command is DNF5. The syntax resembles DNF4, but some subcommands and config-manager changed. Settings live in `/etc/dnf/dnf.conf` or under `/etc/dnf/libdnf5.conf.d/`.

`$ dnf history list`	Lists every install/remove transaction with its number.
`$ dnf history info <id>`	Shows what changed in a transaction.
`$ sudo dnf history undo <id>`	Undoes that transaction (removes what it installed, installs what it removed).
`$ sudo dnf history rollback <id>`	Rolls the system back to the state after that transaction.
`$ dnf provides '*/<file>'`	Finds which package provides a file or command. Example: `dnf provides '*/libssl.so.3'`.
`$ dnf repoquery -l <package>`	Lists the files a package would install, without installing it.
`$ dnf repoquery --whatrequires <package> --installed`	Shows which installed packages depend on it.
`$ dnf repoquery --userinstalled`	Lists packages you installed yourself, not as dependencies.
`$ dnf group list`	Lists package groups.
`$ sudo dnf install @development-tools`	Installs a group (gcc, make, git, etc.). For C/C++: `@c-development`.
`$ sudo dnf reinstall <package>`	Reinstalls a broken package.
`$ sudo dnf downgrade <package>`	Downgrades a package to the previous version in the repository.

`$ dnf download <package>`	Downloads a package as an .rpm without installing it. `--resolve` fetches dependencies too.
`$ sudo dnf distro-sync`	Syncs installed packages with the repository versions; fixes mixed states.
`$ sudo dnf clean all && sudo dnf makecache`	Clears the cache and downloads metadata again.
`$ sudo dnf config-manager setopt max_parallel_downloads=10`	Raises the number of parallel downloads.
`$ sudo dnf install 'dnf5-command(versionlock)'`	Installs the versionlock plugin.
`$ sudo dnf versionlock add <package>`	Locks a package at its current version so it isn't updated. `versionlock list` / `delete`.
`$ dnf needs-restarting -r`	Tells you whether a reboot is needed after updates.
`$ dnf upgrade --security`	Installs security updates only. `dnf advisory list` lists advisories.

> TIP `installonly_limit` sets how many old kernels are kept (default 3).

Repositories: RPM Fusion, COPR, third-party

Repository definitions are in `/etc/yum.repos.d/*.repo`. In DNF5, config-manager uses the `setopt` / `addrepo` syntax.

`$ dnf repolist`	Lists enabled repositories. `dnf repolist --all` shows disabled ones too.
`$ sudo dnf config-manager setopt <repo-id>.enabled=0`	Disables a repository (1 enables it).

`$ sudo dnf config-manager addrepo --from-repofile=<URL>`	Adds a .repo file from the internet.
`$ sudo dnf copr enable <user>/<project>`	Enables a COPR (community build service) repository. Trust is up to you.
`$ sudo dnf copr list`	Shows enabled COPR repositories. `copr remove` removes one.
`$ sudo dnf config-manager setopt google-chrome.enabled=1`	With third-party repositories on, enables the Chrome repository. Then `sudo dnf install google-chrome-stable`.
`$ sudo dnf config-manager setopt fedora-cisco-openh264.enabled=1`	OpenH264 for WebRTC in Firefox. Then `sudo dnf install mozilla-openh264`.
`$ sudo rpm --import <key-URL>`	Imports a repository's GPG signing key.
`$ sudo dnf install rpmfusion-free-appstream-data rpmfusion-nonfree-appstream-data`	Makes RPM Fusion packages show up in Software.

> **WARNING** If COPR and third-party repositories aren't ready for a new release yet, a release upgrade can hit conflicts. Disable them temporarily if needed.

RPM queries

`rpm` queries the installed package database directly. Install with DNF; rpm is ideal for queries and verification.

`$ rpm -qa | sort`	All installed packages.
`$ rpm -qa --last | head`	Most recently installed packages, with dates.
`$ rpm -qf <file-path>`	Tells you which package a file belongs to. Example: `rpm -qf /usr/bin/ls`.
`$ rpm -ql <package>`	Files of an installed package.

`$ rpm -qc <package>`	A package's configuration files.
`$ rpm -qi <package>`	Package info: version, install date, license.
`$ rpm -q --changelog <package> | head -30`	A package's changelog.
`$ rpm -V <package>`	Verifies a package's files; shows changed or missing ones.
`$ rpm -E %fedora`	Prints the Fedora release number; handy in scripts.
`$ sudo rpm --rebuilddb`	Rebuilds a corrupted RPM database.

Release upgrades and firmware

Fedora releases a new version roughly every six months and supports each for about 13 months. Upgrade from Software or from the terminal.

`$ sudo dnf upgrade --refresh`	Fully update the current release first.
`$ sudo dnf system-upgrade download --releasever=<version>`	Downloads the new release's packages. Example: `--releasever=45`.
`$ sudo dnf offline reboot`	Reboots and applies the upgrade in offline mode.
`$ sudo dnf offline log`	Shows logs of previous offline upgrades.
`$ sudo dnf autoremove`	Cleans up packages no longer needed after the upgrade.
`$ sudo dnf install rpmconf && sudo rpmconf -a`	Helps you merge .rpmnew / .rpmsave config files created by the upgrade.
`$ fwupdmgr get-devices`	Devices that support firmware updates.

```
$ fwupdmgr refresh && fwupdmgr update
```

Installs manufacturer firmware updates (LVFS).

> TIP Skipping a release (e.g. 43 to 45) is supported, but skipping more than two releases isn't recommended.

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-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

On Fedora system logs are kept in the systemd journal. There is no /var/log/messages.

`$ 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. `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

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>" ipv4.method manual ipv4.addresses 192.168.1.50/24 ipv4.gateway 192.168.1.1 ipv4.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.
`$ ip -br a`	Interfaces and IP addresses, briefly.
`$ ip route`	Routing table and default gateway.
`$ resolvectl status`	DNS servers (systemd-resolved).
`$ 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 (bind-utils package).

Firewall: firewalld

On Fedora Workstation the default zone is `FedoraWorkstation`; SSH and ports 1025–65535 are open to incoming connections. Rules without `--permanent` are lost on reboot.

`$ sudo firewall-cmd --state`	Whether firewalld is running.
`$ sudo firewall-cmd --get-active-zones`	Active zones and their interfaces.
`$ sudo firewall-cmd --list-all`	All rules in the default zone.
`$ sudo firewall-cmd --permanent --add-service=http`	Opens a service permanently.
`$ sudo firewall-cmd --permanent --add-port=8080/tcp`	Opens a port permanently.
`$ sudo firewall-cmd --permanent --remove-port=8080/tcp`	Closes a port.
`$ sudo firewall-cmd --reload`	Applies the permanent rules.
`$ sudo firewall-cmd --set-default-zone=public`	Switches to a stricter zone (e.g. on public networks).
`$ sudo firewall-cmd --get-services`	Predefined service definitions.
`$ sudo dnf install firewall-config`	Graphical management tool.

SELinux

SELinux limits where a program can reach even if it's compromised. If something says “Permission denied” while file permissions look right, SELinux is usually the cause. The fix is the right label, not turning it off.

`$ getenforce`	Mode: Enforcing / Permissive / Disabled.
`$ sestatus`	Detailed status.
`$ sudo ausearch -m AVC -ts recent`	Shows recent denials (AVC).
`$ sudo sealert -a /var/log/audit/audit.log`	Explains denials and suggests a fix (setroubleshoot-server package).
`$ ls -Z <path>`	Shows a file's SELinux context (label). For processes: `ps -eZ`.
`$ sudo restorecon -Rv <path>`	Resets labels to the default rules. Often needed for files moved with mv.
`$ sudo semanage fcontext -a -t httpd_sys_content_t "/srv/web(/.*)?"`	Adds a permanent labeling rule for a folder; run restorecon afterwards.
`$ sudo semanage port -a -t http_port_t -p tcp 8081`	Lets a service listen on a non-standard port.
`$ getsebool -a | grep <key>`	SELinux on/off options (booleans).
`$ sudo setsebool -P httpd_can_network_connect on`	Turns a boolean on permanently.
`$ sudo setenforce 0`	Switches to Permissive temporarily (for diagnosis only; reverts on reboot).
`$ sudo touch /.autorelabel && sudo reboot`	Relabels the whole file system at boot.

> **WARNING** Instead of disabling SELinux permanently in `/etc/selinux/config`, solve the problem with sealert. For Podman bind mounts use `-v /path:/path:Z`.

Users and permissions

On Fedora sudo rights come from membership in the `wheel` group.

`$ id`	Your user ID and groups.
`$ sudo useradd -m -G wheel <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.

SSH

```
$ sudo systemctl enable --now sshd
```

Turns on the SSH server on this computer (same as Settings > System > Secure Shell).

```
$ 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 run `sudo systemctl restart sshd` .

Btrfs, snapshots and zram

Fedora uses Btrfs by default: the `root` subvolume is mounted at `/` , the `home` subvolume at `/home` , and zstd compression is on. Swap is zram, compressed in RAM instead of on disk.

```
$ sudo btrfs filesystem usage /
```

Real usage according to Btrfs (df doesn't reflect compression).

```
$ sudo btrfs subvolume list /
```

Lists subvolumes.

```
$ sudo btrfs subvolume snapshot -r /
/.snapshots/<name>
```

Read-only snapshot of root (create the `/.snapshots` folder first).

`$ sudo btrfs scrub start /`	Checks data integrity against checksums. Follow with `scrub status /`.
`$ sudo compsize /`	Shows the compression ratio (compsize package).
`$ sudo dnf install snapper btrfs-assistant`	Snapper for automatic snapshots and a graphical manager.
`$ sudo snapper -c root create-config /`	Creates a Snapper configuration for root.
`$ sudo snapper -c root create --description "<description>"`	Takes a snapshot manually. `snapper -c root list` lists them.
`$ sudo snapper -c root undochange <before>..<after>`	Reverts the changes between two snapshots.
`$ zramctl`	zram device, size and compression algorithm.
`$ swapon --show`	Active swap areas.

TIP Timeshift's Btrfs mode requires Ubuntu-style `@` / `@home` naming and doesn't work with Fedora's `root` / `home` layout. On Fedora use Snapper or Timeshift's rsync mode.

TIP To change the zram size, create `/etc/systemd/zram-generator.conf`; the defaults are in `/usr/lib/systemd/zram-generator.conf`.

Kernel, GRUB and booting

Fedora keeps GRUB entries in BLS (Boot Loader Specification) format under `/boot/loader/entries/`.
Use grubby for kernel parameters.

`$ rpm -q kernel`	Installed kernels.
`$ sudo grubby --info=ALL`	All boot entries and their parameters.
`$ sudo grubby --default-kernel`	The default kernel.

`$ sudo grubby --set-default /boot/vmlinuz-<version>`	Changes the default kernel.
`$ sudo grubby --update-kernel=ALL --args="<parameter>"`	Adds a parameter to all kernels. `--remove-args` removes it.
`$ cat /proc/cmdline`	Parameters of the running kernel.
`$ sudo grub2-editenv - unset menu_auto_hide`	Shows the GRUB menu on every boot.
`$ sudo grub2-mkconfig -o /boot/grub2/grub.cfg`	Regenerates the configuration after changes to `/etc/default/grub`.
`$ sudo dracut -f`	Rebuilds the initramfs for the running kernel.
`$ lsmod | grep <module>`	Loaded kernel modules. `modinfo` for details, `sudo modprobe` to load.
`$ mokutil --sb-state`	Whether Secure Boot is on.

Drivers: NVIDIA, codecs, hardware acceleration

These steps require RPM Fusion (free + nonfree).

`$ sudo dnf install akmod-nvidia xorg-x11-drv-nvidia-cuda`	The NVIDIA driver. After installing, wait about 5 minutes for the module to build, then restart.
`$ modinfo -F version nvidia`	Whether the module is built; if it prints a version, it's ready.
`$ nvidia-smi`	GPU status and usage.
`$ sudo kmodgenca -a`	Generates a module signing key for Secure Boot.

`$ sudo mokutil --import /etc/pki/akmods/certs/public_key.der`	Enrolls the key in MOK. Set a password; after rebooting choose “Enroll MOK” on the blue MOK screen and enter the same password.
`$ sudo dnf swap mesa-va-drivers mesa-va-drivers-freeworld`	H.264/HEVC hardware video decoding on AMD.
`$ sudo dnf install intel-media-driver`	Hardware video decoding on Intel (Broadwell and later).
`$ vainfo`	Verifies VA-API support (libva-utils package).
`$ lspci -k | grep -A 3 -E "VGA|3D"`	The graphics card and the kernel driver in use.
`$ sudo dnf install akmod-wl`	Driver for Broadcom Wi-Fi cards.

WARNING Don't install the NVIDIA driver from the .run file on nvidia.com; it breaks on kernel updates. akmod rebuilds the module for every kernel by itself.

Toolbx, Podman, Distrobox

Toolbx installs development tools without cluttering the host, Distrobox gives you other distributions' tools, and Podman runs services. Podman is largely compatible with Docker commands and doesn't need root.

`$ toolbox create`	Creates a Fedora development container matching your release.
`$ toolbox enter`	Enters the container; your home folder is shared and you can use sudo dnf freely inside.
`$ toolbox list`	Lists containers and images. `toolbox rm -f <name>` deletes one.
`$ sudo dnf install distrobox`	Installs Distrobox.

`$ distrobox create -i ubuntu:24.04 -n <name>`	Creates an Ubuntu container; for Arch use `-i archlinux`.
`$ distrobox enter <name>`	Enters it. Inside, `distrobox-export --app <app>` adds the app to the host menu.
`$ podman run -it --rm fedora bash`	Starts a throwaway Fedora 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:Z docker.io/library/nginx`	Runs a service with a mounted folder; `:Z` is for the SELinux label.

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

`$ sudo dnf install podman-compose`	Runs docker-compose files with Podman.

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

Fedora's default `~/.bashrc` automatically reads every file in `~/.bashrc.d/`. Keeping your additions there as separate files keeps things tidy.

`$ mkdir -p ~/.bashrc.d && nano ~/.bashrc.d/aliases.sh`	A separate file for aliases.
`alias update='sudo dnf upgrade --refresh && flatpak update -y'`	System + Flatpak 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="$HOME/bin:$PATH"`	Adds a folder to PATH. `~/.local/bin` is already in PATH on Fedora.
`$ echo $SHELL`	The shell you are using.

```
$ sudo dnf install zsh util-linux-user &&
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` .

Hidden settings with gsettings

All GNOME settings live in the dconf database. gsettings reaches options that aren't in the Settings app.

```
$ gsettings set org.gnome.desktop.interface
color-scheme 'prefer-dark'
```

Dark style ('default' for light).

```
$ gsettings set org.gnome.desktop.interface
clock-show-seconds true
```

Shows seconds on the clock.

```
$ gsettings set org.gnome.desktop.interface
clock-show-weekday true
```

Shows the weekday next to the clock.

```
$ gsettings set org.gnome.mutter center-new-
windows true
```

Opens new windows in the center of the screen.

```
$ gsettings set
org.gnome.desktop.peripherals.touchpad tap-
to-click true
```

Tap to click.

```
$ gsettings set
org.gnome.desktop.wm.preferences focus-mode
'sloppy'
```

The window under the mouse gets focus.

```
$ gsettings set org.gnome.shell disable-user-
extensions true
```

Turns off all extensions at once (troubleshooting). false turns them back on.

```
$ gsettings list-recursively
org.gnome.desktop.interface
```

All keys and values in a schema.

`$ gsettings reset <schema> <key>`	Resets a setting to its default.
`$ dconf dump / > ~/gnome-settings.ini`	Backs up all GNOME settings to a file.
`$ dconf load / < ~/gnome-settings.ini`	Restores the backup; useful for carrying settings to a new install.
`$ sudo dnf install dconf-editor`	A graphical tool to browse every key with descriptions.

Performance and power

Fedora manages power profiles with tuned (through the tuned-ppd compatibility layer); the modes in Settings > Power depend on it.

`$ powerprofilesctl list`	Power profiles and which one is active.
`$ powerprofilesctl set performance`	Switches profile: power-saver / balanced / performance.
`$ tuned-adm active`	The active tuned profile. `tuned-adm list` lists them all.
`$ sudo dnf install lm_sensors && sensors`	CPU and disk temperatures.
`$ sudo dnf install btop nvtop`	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 (fstrim.timer already does it weekly).

```
$ cat /sys/class/power_supply/BAT0/capacity
```

Battery percentage; `upower -i $(upower -e | grep BAT)` gives detailed battery health.

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.** Pick the previous entry in the GRUB menu. If that works, the kernel is the problem; use `sudo dnf history undo` or wait for the next update.

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 gdm
```

Restarts the graphical login manager.

```
$ sudo dnf history undo <id>
```

Undoes the problematic update.

```
$ sudo dnf distro-sync && sudo dnf check
```

Fixes package inconsistencies and reports broken dependencies.

TIP **3. Rescue mode.** In GRUB, highlight the entry and press `e`, add `systemd.unit=rescue.target` to the end of the `linux` line, and boot with Ctrl+X.

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

```
$ sudo mount -o subvol=root /dev/<nvme0n1p3>
/mnt
```

Mounts the Btrfs root subvolume.

```
$ sudo mount /dev/<nvme0n1p2> /mnt/boot &&
sudo mount /dev/<nvme0n1p1> /mnt/boot/efi
```

Mounts the /boot and EFI partitions.

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

Binds the system folders and switches into the installed system; dnf, grub2 and dracut work inside.

`$ sudo dnf reinstall shim-* grub2-efi-* grub2-common`	Repairs the EFI boot loader (inside the chroot, without sudo).

Commands are written for current Fedora Workstation releases (DNF5, GNOME, Ptyxis terminal). Menu names follow the English interface. Commands that start with `sudo` change the system; read what they do before running them.