

DNF system upgrade



fedoraproject.org/wiki/DNF_system_upgrade



🔑 What is DNF system upgrade?

[dnf-plugin-system-upgrade](#) is a plugin for the [dnf](#) package manager which handles system upgrades. It is the recommended command line upgrade method for Fedora 21 and later (Except Atomic Host And Silverblue, which uses rpm-ostree; for that see [Atomic_Host_upgrade](#)).

🔑 What does DNF system upgrade do?

DNF system upgrade can upgrade your system to a newer release of Fedora, using a mechanism similar to that used for offline package updates. The updated packages are downloaded while the system is running normally, then the system reboots to a special environment (implemented as a systemd target) to install them. Once installation of the updated packages is complete, the system reboots again to the new Fedora release.


🔑 How do I use it?

1. **Back up** your important data. Every system change is potentially risky, be prepared. In case you update your workstation, it is also wise to download a [Workstation Live image](#) and make sure your hardware (graphics card, wifi, etc) works well with the latest kernel and drivers.
2. Update your system using the standard updater for your desktop or **dnf** :

```
$ sudo dnf upgrade --refresh
```

(Don't type the **\$** in these commands; that just indicates that you type this at a terminal prompt as a non-root user.)

After updating, we recommend you reboot your computer, especially if you've just installed a new kernel.

- Please note that there is [an issue](#) if you use a non-default plymouth boot theme. If you do, please follow the issue description to make sure your upgrade will not be affected.
 - Double check your DNF configuration in `/etc/dnf/dnf.conf` , if you have done any custom configuration (either manually or via third-party tool), it's recommended to revert it to default before updating and upgrading your system.
3. Install  [dnf-plugin-system-upgrade](#) package:

```
$ sudo dnf install dnf-plugin-system-upgrade
```

4. Download the updated packages:

```
$ sudo dnf system-upgrade download --refresh --releasever=29
```

Change the `--releasever=` number if you want to upgrade to a different system release. Most people will want to upgrade to the latest stable release, which is **29**, but if you're running Fedora 27, you might want to upgrade just to Fedora 28. You can also use **30** for upgrading to Branched or **rawhide** for upgrading to Rawhide (warning: those are not stable releases).

If you are upgrading to Rawhide, you will need to import the rpm gpg key for it. This will be the highest numbered key version in `/etc/pki/rpm-gpg/`. For example if there is a Branched release that is 30, then you should look for a 31, and if there is currently no Branched release, it will be 30.

```
$ sudo rpm --import /etc/pki/rpm-gpg/RPM-GPG-KEY-fedora-31-primary
```

5. If some of your packages have unsatisfied dependencies, the upgrade will refuse to continue until you run it again with an extra `--allowerase` option. This often happens with packages installed from third-party repositories for which an updated repositories hasn't been yet published. Please study the output very carefully and examine which packages are going to be removed. None of them should be essential for system functionality, but some of them might be important for your productivity.

- In case of unsatisfied dependencies, you can sometimes see more details if you add `--best` option to the command line.
- If you want to remove/install some packages manually before running `dnf system-upgrade download` again, it's advisable to perform those operations with `--setopt=keepcache=1` `dnf` command line option. Otherwise the whole package cache will be removed after your operation, and you'll need to download all the packages once again.

6. Trigger the upgrade process:

```
$ sudo dnf system-upgrade reboot
```

This will reboot your machine immediately. The system should boot again into Fedora using the same kernel, but this time, the upgrade process appears on the boot screen.

7. Wait for the upgrade process to complete.

Frequently Asked Questions

How do I report issues that I find with upgrades?

First see [Common F29 bugs](#) or [Common F30 bugs](#) to check if the problem is a very prominent issue we already know of. If it is not there, [search for an existing bug report](#). If you do not see a report that matches your symptoms, you can file a new report from the search page. Please follow the bug reporting instructions mentioned in [this README](#) and in `man dnf.plugin.system-upgrade`.

If you hit issues after upgrade with a specific package, file a bug against the package with which you are having issues.

🔑 Does DNF system upgrade verify the software it runs or installs during upgrade?

Yes. The package signing keys for newer Fedora releases are sent to older Fedora releases in order to allow DNF to verify the integrity of the packages it downloads. You can disable this function with the `--nogpgcheck` parameter if you need to do so for any reason (not recommended, you're then opened to attacks from malicious software).

🔑 Will packages in third party repositories be upgraded?

Yes, if they are set up like regular DNF repositories and do not hard code the repository path. Commonly-used third party repositories usually work fine, but if you attempt to upgrade prior to or soon after an official Fedora release, they may not have updated their repository paths yet, and DNF may be unable to find their packages. This will usually not prevent the upgrade running successfully, though, and you can update the packages from the third-party repository later.

🔑 Can I upgrade from an End of life release?

Note that Fedora strongly recommends against ever running an end-of-life release on any production system, or any system connected to the public internet, in any circumstances. You should never allow a production Fedora deployment to reach end-of-life in the first place.

With that in mind, if you do have an end-of-life release newer than Fedora 20 installed on a system you cannot just discard or re-deploy, you can attempt to upgrade it, though this is a less-tested and less-supported operation. You can try to upgrade through intermediate releases until you reach a currently-supported release, or try to upgrade to a currently-supported release in a single operation. It is not possible to state with certainty which approach is more likely to be successful.

If you attempt to upgrade across more than two releases in one operation, please also read the [next answer](#).

If you have Fedora 20 or earlier installed, you cannot upgrade with DNF system upgrade alone. You must upgrade at least part of the way [using bare dnf or yum](#). You can either upgrade to Fedora 21 that way and then upgrade the rest of the way using DNF system upgrade, or you

can attempt the entire upgrade using bare `dnf` or `yum`. Note this method is in itself not an officially recommended upgrade mechanism. To be frank, any upgrade from Fedora 20 or earlier is very much done 'at your own risk'.

How many releases can I upgrade across at once?

The most common scenario is an upgrade across just one release (e.g. Fedora 28 to Fedora 29). However, for the first month or so after a new release comes out, upgrades from the last-but-one release to that release are 'supported', in the sense that we include this scenario in the [Fedora Release Criteria](#), test it for at least clean installs of supported package sets, and will treat bugs discovered in such upgrades as significant. The [Fedora Release Life Cycle](#) is specifically designed to provide this approximate one month 'grace period' so you can choose to upgrade long-lived systems only once every two releases, rather than having to do it every release.

Around a month after the new release comes out, the last-but-one release goes [End of life](#), at which point the [previous question](#) applies. Still, that upgrade is still pretty likely to work successfully for some time after the release goes end-of-life.

Upgrades across more than two releases are not 'supported', and issues encountered in such upgrades may not be considered significant bugs. Note that any upgrade across more than two releases must by definition be an upgrade from an end-of-life release, and so the [previous question](#) applies here too.

When upgrading across multiple releases, you may find you need to [import the target release package signing key manually](#). Fedora releases usually only have the package signing keys for the next two releases installed (because they go end-of-life before the N+3 release is branched). Before Fedora 22, it was not consistently the case that every release had keys for the next two releases, either. If `dnf` complains about a missing key, this is what you must do.

Can I use DNF system upgrade to upgrade to a pre-release (e.g. a Beta)?

Yes. It should always be possible to attempt such an upgrade. Of course, this function is as subject to temporary breakage as is any other aspect of a pre-release, and generally speaking, the earlier the release in question, the less likely it is to work without problems.

Optional post-upgrade tasks

These are tasks you can do after a successful upgrade. **They are mostly intended for power users. If you are a general user who doesn't use terminal daily, you don't need to worry about this.**

Update system configuration files

Most configuration files are stored in `/etc` . If there are any updates to them and you touched some of those files before, RPM creates new files with either `.rpmnew` suffix (the new default config file), or `.rpmsave` suffix (your old config file backed up). You can search for these files, go through the changes and make sure your custom changes are still included and the new defaults are applied as well. A tool that tried to simplify this is `rpmconf`. Install the package, and then use it as:

```
$ sudo rpmconf -a
```

See more information in its manual page.

Clean up old packages

You can see list of broken packages (unsatisfied dependencies, duplicated packages, etc) using this command:

```
$ sudo dnf check
```

Ideally, there should be no packages listed. If there are some, consider fixing the issues by removing them (or some of them). Always be careful and don't do anything you don't understand well (ask somebody for an advice).

Some packages might stay on your system while they have been removed from the repositories. See them using:

```
$ sudo dnf list extras
```

If you don't use these, you can consider removing them: `dnf remove $(dnf repoquery --extras --exclude=kernel,kernel-*)` . Please note that this list is only valid if you have a fully updated system. Otherwise you'll see all installed packages which are no longer in the repositories, because there is a newer update available. So before acting on these, make sure you have run `sudo dnf update` and generate the list of extra packages again. Also, this list might contain packages installed from third-party repositories for which an updated repository hasn't been published yet. This often involves e.g. RPM Fusion or Dropbox.

You can remove no-longer-needed packages using:

```
$ sudo dnf autoremove
```

but **beware** that dnf decides that a package is no longer needed if you haven't explicitly asked to install it and nothing else requires it. That doesn't mean that package is not useful or that you don't use it. **Only remove what you are certain you don't need** . There's a known bug in PackageKit which doesn't mark packages as user-installed, see [bug 1259865](#). If you use PackageKit (or GNOME Software, Apper, etc) for installation, this output might list even important apps and system packages, so beware.

Resolving post-upgrade issues

Only follow up these steps if you have troubles with your upgraded system. It should not be needed in the vast majority of upgrades.

Rebuilding RPM database

If you see warnings when working with RPM/DNF tools, your database might have gotten corrupted for some reason. It is possible to rebuild it and see if resolves your issues. Always back up `/var/lib/rpm/` first. To rebuild the database, run:

```
$ sudo rpm --rebuilddb
```

Using distro-sync to resolve dependency issues

The system upgrade tool uses distro-sync method by default. If your system stayed partly unupgraded or you see some package dependency issues, you might try to fix it by running another distro-sync manually. This tries to make your installed packages exactly the same version as in currently enabled repositories, even if it meant downgrading some packages:

```
$ sudo dnf distro-sync
```

A stronger variant also allows to remove package for which package dependencies can't be satisfied. Always carefully review which packages are going to be removed before confirming this:

```
$ sudo dnf distro-sync --allowerasing
```

Relabel files with latest SELinux policy

If you see warnings that some actions were not allowed because of current SELinux policy, it might be a case of having some files incorrectly label with SELinux permissions. This might happen in case of some bug or if you had SELinux disabled in some point of time in the past. You can relabel the whole system by running:

```
$ sudo fixfiles onboot
```

and rebooting. The next boot will take a long time and will check and fix all SELinux labels on all your files.