Upgrading and Patching

coursera.org/learn/linux-for-developers/supplement/qRxex/upgrading-and-patching

From time to time, software on the system must be upgraded (or updated) or patched for one or more of the following reasons:

- Incorporation of new features
- Optimization and performance improvements
- Security and bug fixes.

In many operating systems, such patching can be a laborious and error-prone task, which may require frequent reboots. These can be extremely disruptive on server systems, and system administrators have learned to be very cautious when deploying patches.

Linux does not employ a patching model. Rather, it installs complete new packages. For example, on RPM-based systems, the command:

```
1
$ sudo rpm -Uvh libaio-devel-0.3.109-12.el7.x86_64.rpm
```

will put in the newest version; a similar command, using **dpkg**, can be used on Debian-based systems. Even better, depending on the distribution, one of the following commands:

```
1
2
3
$ sudo dnf update libaio-devel
$ sudo zypper update libaio-devel
$ sudo apt-get upgrade libaio-dev
```

will go out and check if an update is available, download it and install it, and, at the same time, update any other installed packages that require synchronization. The commands:

```
4

$ sudo apt-get dist-upgrade
```

will update (and/or upgrade) all packages on the system. Note that Debian-based systems require **update** to first synchronize repository information before updating the entire package system.

In order to avoid large downloads, most recent distributions can use features such as delta rpms, which can download a smaller binary patch file, and then use that to recreate the full rpm. However, when bandwidth is high, the time spent on reconstruction of the full rpm may constitute a net minus.

All distributions have configurable background daemons that run at specified intervals to check for updates, and then either install them automatically, or ask for approval.

The only time you should ever need to reboot a Linux system during the update process is when the kernel itself is updated. This is a feature whose beauty cannot be overemphasized.

The following table lists the basic packaging operations and their **rpm** and **deb**-based equivalents. The **zypper**-based commands are almost identical to the **yum** ones; look at the man page for **zypper** on SUSE-based systems.

Operation	RPM	deb
Install a package	**rpm -i foo.rpm**	**dpkg --install foo.deb**
Install a package with dependencies from repository	**dnf install foo**	**apt-get install foo**
Remove a package	**rpm -e foo.rpm**	**dpkg --remove foo.deb**
Remove a package and dependencies using a repository	**dnf remove foo**	**apt-get remove foo**

Operation	RPM	deb
Update package to a newer version	**rpm -U foo.rpm**	**dpkg --install foo.deb**
Update package using repository and resolving dependencies	**dnf update foo**	**apt-get install foo**
Update entire system	**dnf update**	**apt-get dist-upgrade**
Show all installed packages	**rpm -qa or yum list installed**	**dpkg --list**
Get information about an installed package including files	**rpm -qil foo**	**dpkg --listfiles foo**
Show available packages with “foo” in name	**dnf list foo**	**apt-cache search foo**
Show all available packages	**yum list**	**apt-cache dumpavail foo**
What package does a file belong to?	**rpm -qf file**	**dpkg --search file**

Older RHEL/CentOS Recent Fedora systems used **yum** instead of **dnf**. The basic commands are the same. However, advanced commands can be different or missing. In most cases, if you issue a **yum** command, a warning may be displayed and the equivalent **dnf** command is displayed and carried out.