6.1.2 RPM Package Management

Click one of the buttons to take you to that part of the video.

RPM Package Management 0:00-0:17

In this lesson, we're going to review how you manage RPM packages on a Linux system from the shell prompt. Now, just as with any other operating system, on a Linux system you need to know how to install, repair, and uninstall software from the system.

Install Packages 0:18-4:38

Let's begin by discussing how you go about installing an RPM package from the shell prompt.

Now this is done using the rpm command. You enter rpm -i followed by the package file name that you want to install on the system.

Now, it's important to note that many times when you're running the rpm command, you don't actually have to put in the entire file name. You just put in the name of the package. But in this particular instance, because this particular RPM package has not been installed, RPM doesn't know what it is. We have to actually provide the full file name to that package.

Later on when we're managing that installed software with the rpm command, we can just call it by its packaged name, instead of the full file name--which in this case is gftp. If I wanted to install the gftp package on my system, I would run rpm -i then the full file name of the gftp package.

Now, when I run this command the software will be installed on my system from the package, and then the application is ready to be used. There's one key weakness with the rpm -i command, and that is the fact that it doesn't provide a whole lot of feedback to you as the system administrator as it installs the package.

Some system administrators don't care about that. They're just fine. But some system administrators, like me, prefer to see some kind of visual display during the installation process so we can see where things are at.

To do this, you could include the -h and the -v options with the -i option. -h tells the RPM utility to print hash marks (#) on the screen as a progress indicator. Then the -v option tells RPM to operate verbosely, meaning that it's going to put the output on the screen.

One useful feature of the RPM utility is its ability to calculate dependencies for you. You need to understand that a given Linux software package may be dependent upon another package already being installed on the system first before it can work properly.

If this is the case, then the dependent package is called a dependency. When you run the RPM utility, it's going to take a look at the package that you want to install and automatically check your system to see if you have all of the dependent packages already installed.

If you don't, it's going to generate an error and it's going to prompt you to install the necessary packages that are required for that particular software package to work.

If you want to just check dependencies for a particular RPM without actually installing the software, you can use the --test option with the -i option with the RPM utility. An example of doing that is shown here: rpm -i --test and then the name of the RPM package that we want to test.

Even though we specify a -i in the command, it doesn't actually install the package. It just queries the package and says, "Hey, what packages are you expecting to be on the system, so I can go check and make sure they're there before we actually install you?"

If there are any failed dependencies, they'll be printed on the screen, which allows you to then go ahead and install the necessary software first.

If you're crazy, you can actually skip the dependency test during the installation process. You can do this by entering rpm -i --nodeps. The nodeps option turns off dependency checking. If you do this, then that package is going to be installed even if the software that it's dependent upon isn't present.

This is usually not a good idea, and I can honestly say, I don't think I've ever done this because it just doesn't make sense. There may be some situation where you need to install a package without dependencies being installed first. Then that's the option that you can use.

After running rpm, you can then use the installed software. Some packages will create a menu item or icon somewhere in your KDE, or known desktop environment. You use that to launch the application.

Other packages--especially if it's a daemon--will have to be started form the shell prompt. In this example right here, I have used the gftp command to load the gftp client. When I run the command at the shell prompt, the application's graphical user interface is displayed, and I can use it to connect to an ftp server.

Uninstall Packages 4:39-4:58

With this in mind, let's take a look at how you uninstall packages from the Linux system. That can be done using the rpm command as shown here. You enter rpm using the -e option, which stands for erase. The -e option tells the RPM utility to remove or erase that package from the system.

Erase a Package 4:59-7:26

Now just a minute ago we installed the gftp package with the rpm -i command, now we're going to erase it from the system using the rpm -e command. Be aware that when you use RPM to erase a package from the system, it's going to check for dependencies during the uninstall process.

If other software is currently installed on the system that is dependent upon the package that you're trying to remove. An error message will be displayed and it will list all those dependent packages. Before you can uninstall the package you're trying to get rid of, all of the dependent packages are going to have to be removed first.

With this in mind, let's discuss how you update installed software on your system. You can do that with the RPM utility. This time using the -U option--not lower case u, but capital U. This will cause RPM to remove any existing packages on the system and then install a newer version of that package.

In this case, we are updating the gftp package on the system. If I had an older version of gftp already on the system, then RPM will uninstall it and then install the newer version.

Now, here's a quick tip. If you use the -U option trying to update a package, but that package hasn't actually been installed on your system yet, you're okay. RPM will just go ahead and install the package.

In fact, if I'm using RPM to install a package on the system, I actually don't use the -i option very often. Usually I use -U, regardless of whether I'm installing or updating a package. Because if the package isn't there, it'll automatically install it for me. Notice that with -U you can use the -h and -v options, just like with the -i option, to display a progress indicator on the screen.

In addition to installing and upgrading packages on your system, you can also use RPM to query packages on the system. That's done using the -q option with the rpm command. This allows us to view information about installed packages on the system.

Be aware that when you use -q, you have to specify exactly what it is that you want to query. In this example, I've run the rpm -qi command.

When I do this, I specify a package name. When I do, detailed information about that package is displayed on the screen. In this case, I entered rpm -qi and then the name of the package, gftp, and I see all this information about that package.

Use whatrequires 7:27-9:29

You can also use the whatrequires option with the -q option with the rpm command to display a list of packages that require a specified package. In this example, I've entered rpm -q --whatrequires and then iptables. When I do, it displays a list of all the packages on the system that require the iptables package to be installed on the system first.

You can also use the -l option to display a list of files that are included within a particular RPM package. You can use the --provides option with the -q option to display the functionality that the particular package provides. You can use the --requires option with the -q option to display the functionality that's required by the specific package.

Another useful option is the --whatprovides option. When used with -q, it is used to determine what package file will provide a particular requirement. In this case I'm saying what provides iptables? It says, well, it's the iptables package, and here's the file name.

In addition to querying packages, the RPM utility can also be used to verify packages on your system. This is really useful because software can get corrupted, deleted, or otherwise messed up after it's been installed on your system. To do this we use the -V option. Then we can make sure that's everything's working the way it's supposed to.

Now, we could use the -V option to verify a single package by entering rpm -V and the name of the file name followed by the name of the package. You can also use -V to verify all of the installed packages on the system. To do that, we just use rpm -Va.

When you use the -V option, and the rpm command then returns no output, it means no errors were found--everything is cool. If errors are found, then the error message is displayed on the screen.

As you can see here on this particular system, there were several different errors found. We see the error code here, and then the name of the file name in question.

Find Errors 9:30-11:36

The error messages use the syntax shown on this slide. We have the error code itself and then a c here to indicate whether it's a configuration file or not. Then the name of the file from the package that has the problem.

Here you can see a key to this error code. S indicates there's a problem with the file size. M indicates a problem with the files mode--the permissions that are assigned to it. 5 indicates there's a problem with the MD5 checksum of a file.

D indicates there's a problem with the file's revision numbers. L indicates there's a problem with the file's symbolic link. U indicates there's a problem with the file's ownership. G indicates there's a problem with the files owning group. T indicates there's a problem with the time stamp on a file.

Anytime one of these letters is represented by a period, such as right here, it indicates that there is no problem with that particular parameter. However, if there is a letter displayed, that indicates that a problem was found for that particular specified file.

In this example right here, we have a file called /etc/hba.conf. We have an S, which indicates there's a problem with the size of the file. We have a 5, which indicates there's a problem with the MD5 checksum of that file. A T, which indicates there's a problem with the time stamp on that file.

Before we go any further I do want to point out that you'll see that most of these files have a c next to them indicating that they're configuration files. These are configuration files that came from a particular package that's installed on the system.

We actually don't get real uptight if rpm -Va returns an error for a configuration file. Why? Because configuration files get modified all of the time. Therefore, the configuration file that's installed on your system is not going to match up with the configuration file in the package.

Hence, we have all the size, and MD5, and time stamp errors that we're seeing for config files. That's because somebody went in with a text editor and changed them.

Repair Packages 11:37-12:17

However, if a different type of file--such as an executable--fails to verify, then we need to be a little bit more concerned. In this situation, based on our output from the rpm -V command, we may need to repair and install the package.

You can use RPM to do this too. It's done using the -U option with --replacepkgs.

For example, here we decided that we needed to reinstall the gftp package. We ran rpm -U --replacepkgs and then the RPM file name. When I do this, this will force RPM to install the gftp package from this original RPM, even though it's already installed on the system.

Summary 12:18-12:30

That's it for this lesson. In this lesson we talked about managing Red Hat packages. We talked about installing packages, uninstalling packages, updating packages, querying packages, and then we ended this lesson by talking about how to verify packages.
