6.1.1 Red Hat Package Manager (RPM)

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

Red Hat Package Manager 0:00-0:33

As a Linux system administrator, you need to know how to install and manage software on a Linux system. If you're already familiar with installing software on, say, a Windows system, you're probably thinking, "Installing software? How hard could that be?" On a Windows system, you run the file, you click, next, next, next, next, finish, right?

Well, actually, installing software on Linux can be somewhat challenging if you're coming to Linux from Windows. It's not that installing software on Linux is any more difficult than it is on other operating systems, it's just that it's very different.

Role and Function of Software Packages 0:34-4:50

If you already have experience installing applications on, say, Windows, then you need to momentarily shelve everything you know about installing software and be prepared to approach the process from an entirely different perspective with Linux. There just aren't enough similarities between Windows and Linux. If you let go of your Windows way of doing things and learn how to do things the Linux way, you're going to be just fine.

When you're installing software on a Linux system, you essentially have two different options. The first option is to install a pre-compiled application or service from a software package. Software packages contain everything you need in one nice little unit.

The package contains the executables that you need, contains the supporting files that you need, such as configuration files, and so on. Everything's been pre-packaged, pre-compiled, and configured for a specific hardware architecture, or possibly even Linux distribution.

The second option you have is to actually install the application or service from source code. When you install an application or service from source, what you have to do is actually take a compiler on your system, and then compile that application source code into an executable that will run on your hardware and on your distribution.

In this lesson, we're going to focus on the first option: installing software from packages. With package software, the source code is already pre-built, and it's pre-configured for a particular system architecture and, in many cases, a specific Linux distribution.

To install software from a package, you have to have a package manager installed. Many distributions--such as openSUSE, Fedora, and so on--use the Red Hat Package Manager, or RPM, to install and manage RPM packages on the system. Other distributions, such as Ubuntu, use the Debian package management system. In this lesson, we're just going to focus on RPM.

Regardless of which package manager you're distribution uses, the package manager performs a similar set of tasks, and these are listed here.

First of all, the package manager is used to install software. It's used to update software that has already been installed. And it's used to uninstall software that you don't want on the system anymore. It's used to query installed software to see if it's installed and if it's functioning properly, and it's also used to verify the integrity of installed software.

To keep track of all this information, RPM stores information about the packages installed on your system in the RPM database file, which as you see here is /var/lib/rpm. Understand that whenever you install a new package on your system, whenever you update a package on your system, or whenever you uninstall a package on your system, the appropriate change is made inside of this database.

Before we go on, I do need to point out that on very rare occasions it is possible for this database file to get corrupted. I've had it happen only to me once, but when it does happen, it is a problem. It makes it, basically, so that you can't install, update, or uninstall anything off of the system.

If this happens, however, you can fix it. It's actually really easy to fix. All you have to do to rebuild the database is run this command here: 'rpm --rebuilddb'. When I had this problem, I ran this command. It just took a few minutes, and the database was fixed and--miracle of miracles--I was able to manage applications and services on my Linux system again.

The good news is RPM packages are actually pretty easy to install. Basically, most RPM packages are downloaded as a single file, and you process that single file to install the application or service. You don't have to compile anything from source code, and you do this using the RPM command at the shell prompt.

Before we talk about how to do that, you need to understand the naming convention used by RPM packages. It frankly is a little confusing. Okay, maybe not a little--quite a bit confusing. But it's important because RPM packages are built for, first of all, a specific CPU architecture. Sometimes they're built for even a specific Linux distribution.

Name Conventions That Are Used With RPM Packages 4:51-8:15

Because RPM packages come pre-compiled for specific hardware architectures, this means that multiple versions of the same RPM packages may be available in order to accommodate different platforms. You may even see multiple versions of the same RPM package for different Linux distributions. We'll talk about that in a second.

The important thing to remember is that if you're downloading and installing an RPM package, you need to make sure that you get the correct version for your system, and you can do this by checking the name of the RPM package.

Here's an example of an RPM package name here. This is for a utility called gftp. It's used to install a simple, graphical FTP client on the Linux system. Different parts of this file name are used to indicate different information about this particular package. First, we have the package name right here. This part of the file name simply identifies the name of the package. In this case, it's gftp.

Next, we have the version number. This part of the file name specifies the version of the software in the package. In this case, the software version number is 2.0.19. Next, we have the release number. This part of the file name indicates the current release of the software version number.

Understand that on occasion, errors are encountered in the process of creating an RPM package. When the package is released and made available for the same version of the software that just fixes a packaging error, this release number right here is incremented. In this example, the release number is 12.

Next, we have the distribution field. This field is optional. Not all packages will have it. But if it does, then this field indicates that this package has been compiled for a very specific Linux distribution. In this case, the distribution is fc21. This indicates that this package was compiled specifically for Fedora Core version 21.

If a package is distribution specific, you may see other values in this field, such as RHL, which specifies that the package is intended for Red Hat Linux. Or you might see SUSE, which specifies that the package has been compiled for a SUSE Linux system.

Then, finally, we have the CPU architecture. This part of the file name identifies which CPU that the software within this package has been compiled to run on. In this example, the CPU architecture is X86_64, which indicates that the software within this package is designed to run on 64-bit X86 CPUs.

Other values might be specified in this field depending upon the package. For example, if it says i386, that means the software will run on any Intel 80386 or later CPU. It's important to note that these are forward looking, meaning that if it says i386, it doesn't mean that it only runs on a 386 processor. I don't think you could even find a 386 processor anymore.

It just means that it will run on any processor as old as a 386. Anything newer should be just fine as well. i586 specifies that the software will run on an Intel Pentium or later. i686 means that the software will run on a Pentium 4 or later. You may also see no architecture, which specifies that the package is not architecture dependent; it will run on any CPU.

How to Download Packages 8:16-9:26

RPM packages are usually located on your distribution installation DVD, and they may also be available from a variety of websites on the internet. However, the best locations for obtaining RPM packages are the package repositories made specifically for your Linux distribution.

These packages have been configured specifically to run on your particular distribution. For example, if you're running a Fedora system, then you can view your package repository locations in this file: /etc/yum.repos.d/fedora.repo.

In fact, you can use the yumdownloader utility that you see right here to download a particular package directly from one of these repositories, which is really nice, because you don't have to use Google to try to find the package.

You don't have to wonder whether you're getting the right version, whether you're getting the right architecture. If you use yumdownloader, followed by the name of the package that you want to pull down, it will download that package file from the repository that is pre-configured for your distribution. It eliminates a lot of confusion and it saves a lot of time.

How to Verify a Downloaded Package 9:27-10:38

After you download a package, it's a really good idea to use the RPM command to check the authenticity of that package. Doing this lets you know whether or not the package you just downloaded came directly from the organization responsible for maintaining that software or whether it has been altered or corrupted in some way during transit.

It could possibly have even been captured midstream, altered by a third party, and loaded with malware--and you don't want that. The way you check the authenticity of the package is to use the rpm checksig command. You run 'rpm --checksig' followed by the file name of the

package.

In this case, we're checking the gftp package. When you do, the RPM command will check the digital signature of the package. Notice in the output of this command, it says that it is OK. This indicates that the package hasn't been altered by anyone since it was created.

However, if it were corrupted in some way or if it were tampered with in some way in transit, then the digital signature wouldn't match anymore, and you would get an error message right here instead, letting you know that you really shouldn't use that package. You need to go get another copy that is intact.

That's it for this lesson.

Summary 10:39-10:54

In this lesson, we introduced you to package software and the Red Hat package manager. We talked about the role and function of software packages. We talked about the naming conventions that are used with RPM packages. We talked about how to download packages. Then, we ended this lesson by talking about how to verify that a package you downloaded hasn't been corrupted.
