Install and Update Software

Linux #redhat #packages

Install and Update Software Packages

- Software is any application that you run on your computer
- Package is a container that contain the software related programs, files and executables

We will learn…

- ✓ Linux package management using `yum`/`dnf` and `rpm` command
  - `yum` command is the older command that has been deprecated, not fully, the newer version is `dnf`.
- ✓ System update and patch management (yum `update` vs. `upgrade`)
- ✓ Advance package management

System Updates and Repos

- Repos stands for "Repositories"
- There are two major commands that are used in system updates and repos installation.
- `yum`/`dnf` (CentOS), `apt-get` (other Linux)
  - The `yum` command installs a package from the repositories that are defined in your Linux system.
  - Those repositories, those configuration files have the URL link for repositories. So when you run `yum` it's telling the system to go online with this URL and get the package that I'm asking you to install.
  - `yum` does need an internet connection to go online and install a package to your system.
    - You would like to prove your internet connection by doing a ping to any website or server.
  - In some companies internet access is not allowed. In that case repos are setup locally on another server within the same network.
  - `yum`/`dnf` is a very smart tool. It downloads, it installs, it verifies, and it also cleans up your download so your space is used more efficiently (it does not keep the older downloads in your system)
- `rpm` (Red Hat Package Manager)
  - `rpm` was first developed by Red Hat and is still in use by Red Hat, but nowadays many other Linux distributions have carried over the same `rpm` command.
  - `rpm` is used when you already have a package downloaded in your system and then you could install it locally by running the command `rpm`. Whereas `yum` does all the things for you (downloads and installs the package).

Download and Install with `yum`/`dnf`

Example using `dnf` command:

```
[root@localhost ~]# dnf install ntp
```

- Note we are root users.

- When you run this command your system will go outside to the mirrors or whatever the URL has been specified and it will look for the `ntp` package.
- Before it goes to look outside (online), it looks for within itself, within its own system and see whether the `ntp` package has been installed already or not.

Output:

```
Updating Subscription Management repositories.
Last metadata expiration check: 0:03:43 ago on Mon 24 Jun 2024 01:28:19 PM CST.
No match for argument: ntp
Error: Unable to find a match: ntp
```

- In my case the `ntp` package is not findable.
- If the package was already installed it will let us know, as well as the version installed, and won't complete a new installation action.

Example using `dnf` command:

```
[root@localhost ~]# dnf install httpd
```

- `httpd` is the Apache package.

Output:

```
Updating Subscription Management repositories.
Last metadata expiration check: 0:08:47 ago on Mon 24 Jun 2024 01:28:19 PM CST.
Dependencies resolved.
================================================================================================================
===============
 Package                     Architecture    Version                Repository
Size
================================================================================================================
===============
Installing:
 httpd                       x86_64          2.4.57-8.el9           rhel-9-for-x86_64-appstream-rpms
52 k
Installing dependencies:
 apr                         x86_64          1.7.0-12.el9_3         rhel-9-for-x86_64-appstream-rpms
126 k
 apr-util                    x86_64          1.6.1-23.el9           rhel-9-for-x86_64-appstream-rpms
97 k
 apr-util-bdb                x86_64          1.6.1-23.el9           rhel-9-for-x86_64-appstream-rpms
14 k
 httpd-core                  x86_64          2.4.57-8.el9           rhel-9-for-x86_64-appstream-rpms
1.5 M
 httpd-filesystem            noarch          2.4.57-8.el9           rhel-9-for-x86_64-appstream-rpms
15 k
 httpd-tools                 x86_64          2.4.57-8.el9           rhel-9-for-x86_64-appstream-rpms
87 k
 redhat-logos-httpd          noarch          90.4-2.el9             rhel-9-for-x86_64-appstream-rpms
18 k
Installing weak dependencies:
 apr-util-openssl            x86_64          1.6.1-23.el9           rhel-9-for-x86_64-appstream-rpms
17 k
 mod_http2                   x86_64          2.0.26-2.el9_4         rhel-9-for-x86_64-appstream-rpms
167 k
```

```
 mod_lua                    x86_64          2.4.57-8.el9            rhel-9-for-x86_64-appstream-rpms
61 k

Transaction Summary
================================================================================================================
===============
Install  11 Packages

Total download size: 2.2 M
Installed size: 6.0 M
Is this ok [y/N]:
```

- It will throw a list of the things about to be installed and it will ask for confirmation (type y for yes and N for no)

Check if a package has already been installed

- Run the command `rpm -qa`
  - Remember the `-q` option stands for "query".
  - The `-a` option stands for "all" and specifies to query all the package that I have installed in my system.
  - It will give you every single package.
- To count the total number of packages run `rpm -qa | wc -l`
  - In this case `wc -l` is counting the lines of `rpm -qa`
- To look for a certain package run `rpm -qa | grep bind`
  - This line will look for the package named `bind` inside the `rpm` query.

Example using `rpm` command:

```
[root@localhost ~]# rpm -qa | grep bind
```

- The `bind` package is a DNS package.

Output:

```
bind-license-9.16.23-18.el9_4.1.noarch
bind-libs-9.16.23-18.el9_4.1.x86_64
bind-utils-9.16.23-18.el9_4.1.x86_64
```

- There are bind libraries and utilities but there is no package with just the name "bind", so you can go ahead and install it using the `dnf` command.
  - Depending on the speed of your computer and how fast it is connected to the internet is what matters in doing the download and install process rapidly.
  - First `dnf` downloads a package, and after completing that process it will begin to install the package, once installed it verifies it and confirms that it has installed the package. You now know how to verify it manually by running the command `rpm -qa | grep bind`

Install with `rpm`

- To install a package you have to use the command

Example using `rpm` command:

```
[root@localhost ~]# rpm -ihv /tmp/package.rpm
```

- Note that the location of the `rpm` package has to be specified after the options.
- The `-i` option stands for "info" and displays package information, including name, version, and description.
- The `-h` option stands for "hash" and prints 50 hash marks as the package archive is unpacked.
- The `-v` option stands for "verbose" and prints verbose information.
- When you run this command it will install that package for you.

Remove a package with `rpm`

Example using `rpm` command:

```
[root@localhost ~]# rpm -e bind
```

- Let's say we wanted to remove the `bind` package previously installed.
- You have to specify the whole the name of the package you want to remove, you can use the Tab function to find the package you are looking for while you are typing or just copy and paste the whole name of the package from `rpm -qa.

Remove a package with `yum`/`dnf`

Example using `dnf` command:

```
[root@localhost ~]# dnf remove bind
```

- Attempting to remove the package named "bind" using the `dnf remove` command.

Output:

```
...
================================================================================================
===============
 Package                 Architecture  Version                     Repository
Size
================================================================================================
===============
Removing:
 bind                    x86_64        32:9.16.23-18.el9_4.1       @rhel-9-for-x86_64-appstream-
rpms        1.4 M
Removing unused dependencies:
 bind-dnssec-doc         noarch        32:9.16.23-18.el9_4.1       @rhel-9-for-x86_64-appstream-
rpms         33 k
 bind-dnssec-utils       x86_64        32:9.16.23-18.el9_4.1       @rhel-9-for-x86_64-appstream-
rpms        420 k
 python3-bind            noarch        32:9.16.23-18.el9_4.1       @rhel-9-for-x86_64-appstream-
rpms        180 k
 python3-ply             noarch        3.11-14.el9                 @rhel-9-for-x86_64-appstream-
rpms        430 k
```

```
Transaction Summary
================================================================================================================
===============
Remove  5 Packages

Freed space: 2.5 M
Is this ok [y/N]:
```

- It will show you what is about to be removed and ask you to confirm the action.

System Upgrade/Patch Management

There are two types of upgrades that happen in Linux operating system.

- Two types of upgrades
  - Major version = 5, 6, 7
    - The Major version cannot be upgraded through the `yum/dnf` command.
    - How would you update the Major version? You would have to back up the entire server, build a new server from scratch and then transfer the files. There are a few other ways that you could do it, but it is not as convenient as the Minor version.
  - Minor version - 9.3, 9.4
    - The Minor version (decimal point number) can be upgraded through `yum/dnf update` command.
- To know which version of Red Hat Linux you are running you can run: `cat /etc/redhat-release`

Example using `cat` command:

```
[user@localhost ~]$ cat /etc/redhat-release
```

- Checking the version of RHEL we currently run.

Output:

```
Red Hat Enterprise Linux release 9.4 (Plow)
```

Example using `dnf` command:

```
[root@localhost ~]# dnf update -y
```

- The `-y` option is telling the operating system that every time it prompts the user to install a certain package it won't ask the user, it will just install it.

Output:

```
...
Upgraded:
...
  python-unversioned-command-3.9.18-3.el9_4.1.noarch       python3-3.9.18-3.el9_4.1.x86_64
  python3-idna-2.10-7.el9_4.1.noarch                       python3-libs-3.9.18-3.el9_4.1.x86_64
  python3-perf-5.14.0-427.22.1.el9_4.x86_64                qemu-guest-agent-17:8.2.0-11.el9_4.3.x86_64
  samba-client-libs-4.19.4-105.el9_4.x86_64                samba-common-4.19.4-105.el9_4.noarch
```

```
  samba-common-libs-4.19.4-105.el9_4.x86_64                 selinux-policy-38.1.35-2.el9_4.2.noarch
  selinux-policy-targeted-38.1.35-2.el9_4.2.noarch          sos-4.7.1-3.el9.noarch
Installed:
  composefs-1.0.3-2.el9.x86_64                            composefs-libs-1.0.3-2.el9.x86_64
  grub2-tools-efi-1:2.06-80.el9_4.x86_64                  grub2-tools-extra-1:2.06-80.el9_4.x86_64
  kernel-5.14.0-427.22.1.el9_4.x86_64                     kernel-core-5.14.0-427.22.1.el9_4.x86_64
  kernel-modules-5.14.0-427.22.1.el9_4.x86_64             kernel-modules-core-5.14.0-
427.22.1.el9_4.x86_64

Complete!
```

Update vs. Upgrade

- `upgrade` = deletes packages.
  - It will delete old packages with a previous version and it will update it with the newer version package.
- `update` = preserve.
  - It keeps the old packages.
  - Sometimes used in case some of the application don't compile or don't run with the newer version of the packages.
- Please note the `yum/dnf update/upgrade` will not update if your system is already up to date.
- Make sure your system can go outside (have internet connection), by pinging any server.
  - `yum/dnf update/upgrade` will download your packages or update your packages

Advance Package Management

- Install packages
  - The best way to do this is through the `yum`/`dnf` command. But what if you already have the package and you just want to install it?
- Upgrading
- Deleting
- View package details information
- Identify the source or location information
  - If you have a file or if you're running a command, and you wanted to know which package that command belongs to.
- Package configuration files
  - All the package configuration files are by default `/etc/`. but then it goes into subdirectories.

Install packages

In this practice we will install the Korn shell. This is a shell, just like a bash or C shell.

1. Check if we have the package already installed in the system by running `rpm -qa | grep ksh`.
   - `ksh` is the name of the package for "Korn Shell"
2. Install the `ksh` package,

Example using `dnf` command:

```
[root@localhost ~]# dnf install ksh*
```

- The `*` sign specifies to also install anything that is related to Korn shell.

Output:

```
Updating Subscription Management repositories.
Last metadata expiration check: 0:06:26 ago on Tue 25 Jun 2024 05:39:41 PM CST.
Dependencies resolved.
================================================================================================================
===============
 Package          Architecture        Version                    Repository
Size
================================================================================================================
===============
Installing:
 ksh              x86_64              3:1.0.6-3.el9              rhel-9-for-x86_64-appstream-rpms
885 k
...
Total download size: 885 k
Installed size: 3.0 M
Is this ok [y/N]: y
```

3. You can check if the package was installed by running `rpm -qa | grep ksh` again.
4. If you wanted to remove or uninstall the package at this point you can run `dnf remove ksh*`
    - Remember to also use the `*` sign to also remove anything related with the package.

- This is the first way and most preferred way to install and download a package, as well as to remove it.

What if you do not have internet access or if you do not have a local repository?

- You have to know the exact location how to download a file and the run the `rpm -ihv` command to install the package.
    - To get this file that will help you download your desired packages you can google the download file. For example, search for "download ksh for RHEL 9". And follow this steps:
        1. Go to the ksh package page.
        2. Look for the `.rpm` package. Sometimes also referred as binary package.
        3. Double click on the blue-highlited link.
        4. Now click on Copy link address.
        5. On the Linux machine you want to install the package run: `wget [URL]`

For example:
Example using `wget` command:

```
[root@localhost ~]# wget https://centos.pkgs.org/9-stream/centos-appstream-x86_64/ksh-1.0.6-
3.el9.x86_64.rpm.html
```

- `wget` is a non-interactive network downloader.
- In this example the link of the ksh package for CentOS is being used.
- How can I download this package if I don't have internet access?
    - You can go ahead and download in the machine that already has internet connection, you download and then you get FTP over or whichever way is preferred for you.

Output:

```
100% [==>] 906,088                                                  282KB/s in 3.1s
2018-08-11 17:51:57 (282 KB/s) ksh-20120801-137.el7.x86_64.rpm' saved [906088/906088]
```

- The package has been downloaded. Where did it download?
  - You can just check `pwd`
  - The `wget` command will install whatever the link says in the current working directory.
  - Remember you can check this by running `ls -ltr`

What if you wanted to verify the information of the package?

Example using `rpm` command:

```
[root@localhost ~]# rpm -qi ksh
```

- The `-i` stands for "information" specifies to give information about the package.
- The name of the package has to be specified.

Output:

```
Name        : ksh
Epoch       : 3
Version     : 1.0.6
Release     : 3.el9
Architecture: x86_64
Install Date: Tue 25 Jun 2024 08:29:01 PM CST
Group       : Unspecified
Size        : 3127621
License     : EPL-1.0
Signature   : RSA/SHA256, Thu 22 Feb 2024 10:00:43 AM CST, Key ID 199e2f91fd431d51
Source RPM  : ksh-1.0.6-3.el9.src.rpm
Build Date  : Sun 11 Feb 2024 04:07:17 PM CST
Build Host  : x86-64-01.build.eng.rdu2.redhat.com
Packager    : Red Hat, Inc. <http://bugzilla.redhat.com/bugzilla>
Vendor      : Red Hat, Inc.
URL         : http://www.kornshell.com/
Summary     : The Original ATT Korn Shell
Description :
KSH-93 is the most recent version of the KornShell by David Korn of
AT&T Bell Laboratories.
KornShell is a shell programming language, which is upward compatible
with "sh" (the Bourne Shell).
```

- This information is about the specific package that has been installed in your system.
- If you want to delete the package using the `rpm` command you can run: `rpm -e ksh`

How to list the configuration file of a package

Example using `rpm` command:

```
[root@localhost ~]# rpm -qc ksh
```

- The `-c` option stands for "configfiles" and specifies to only include configuration files.
- The name of the package is required.

Output:

```
/etc/binfmt.d/kshcomp.conf
/etc/kshrc
/etc/skel/.kshrc
```

- This is the list of all the configuration files
  - Package configuration files are always in the `/etc/` directory, but then once you're in `/etc/` directory there are thousands of files, thousands of directories. So where do we have to go exactly for that specific package? For that you have to find out the configuration file and you can find this by running the previous command

Identify the source or location information

- In this example we will use the command `ksh` as we previously installed the ksh package and we know where it comes from but imagine we do not know where the command actually comes from.
- If you want to know which package this command belongs to. For that, you first have to know the path of that command, then you can to run the command `rpm -qf [command path]`.

Example using `which` command:

```
[user@localhost ~]$ which ksh
```

- `which` shows the full path of (shell) commands.

Output:

```
/usr/bin/ksh
```

- The full path of the `ksh` command is `/usr/bin/ksh`

Example using `rpm` command:

```
[root@localhost ~]# rpm -qf /usr/bin/ksh
```

- Wanting to know the package to which the `ksh` command belongs to
- The `-f` option stands for "file" and specifies to query package owning installed.

Output:

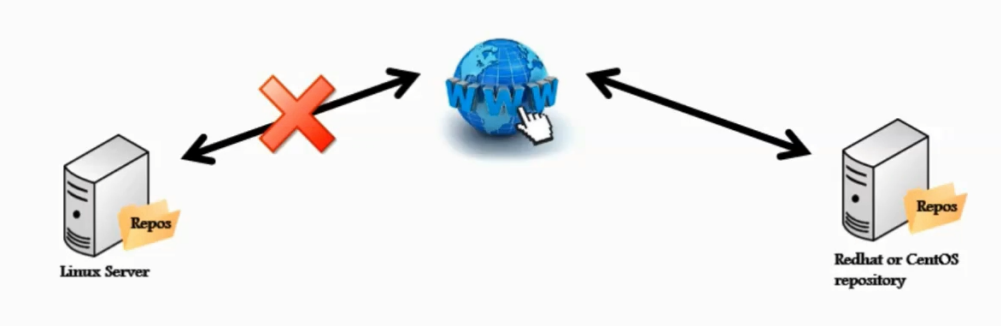
```
ksh-1.0.6-3.el9.x86_64
```

- This is the package to which the `ksh` command belongs to.
- If you remove this package then you will not be able to run the command `ksh`.\

Create local repository from DVD or ISO image

What is a repository?

- A repository is something where all your packages are stored and then you could download and install the package that you wish to.
- You do need internet access to go outside of a repository and download and install the `.rpm` file. But what if you do not have internet access? That's where the local repository comes in.



Before creating a repository

- To exemplify this, first mount a disk ISO image to your VM or Linux machine.
- To verify it was mounted you can run `df -h` command. Remember from Monitor and manage Linux processes. (it will list all the drives of the OS)

Example using `df -h` command:

```
[user@localhost ~]$ df -h
```

Output:

```
Filesystem            Size  Used Avail Use% Mounted on
devtmpfs              4.0M     0  4.0M   0% /dev
tmpfs                 5.1G     0  5.1G   0% /dev/shm
tmpfs                 2.1G  9.2M  2.1G   1% /run
/dev/mapper/rhel-root  17G  7.9G  9.1G  47% /
/dev/sda1             960M  411M  550M  43% /boot
tmpfs                 1.1G  100K  1.1G   1% /run/user/1000
/dev/sr0              991M  991M     0 100% /run/media/mmarin/RHEL-9-4-0-BaseOS-x86_64
```

- Look the last line. We have added an ISO image for RHEL version 9.4.
- We know it is mounted into `/run/media/mmarin/RHEL-9-4-0-BaseOS-x86_64`.
  - If it is not mounted automatically then you could run a command `mount /dev/cdrom...` and wherever you wanted to be mounted.
- So where are you going to put all of your packages? Create a directory.
  - Travel to `/` directory and create a directory using the `mkdir` command, you can name this directory whatever you want.

Example using `cd, mkdir` command:

```
[root@localhost ~]# cd /
[root@localhost /]# mkdir localrepo
```

- The "localrepo" directory has been created.

- Now use the `cp` command to copy all the contents from the disk image to this directory.

Travel to the disk image directory
Example using `cd` command:

```
[root@localhost ~]# cd /run/media/mmarin/RHEL-9-4-0-BaseOS-x86_64/
```

- Be careful: Linux does not like spaces. If the name of the ISO image contains spaces in it the path will be different.
  - It is a good practice to when at the time of typing the path of the disk image you hit the Tab key to auto-complete the path.
- Now you can run `ls -ltr` to see the contents of the disk.

Travel to the `Packages/` directory
Example using `cd` command:

```
[root@localhost RHEL-9-4-0-BaseOS-x86_64]# cd Pacakges/
```

- This example is erroneous since RHEL images do not contain a `Packages/` directory but most other Linux distros do, so it will just be used as an example.
- If you do `ls -ltr` you will see a bunch of `.rpm` packages

If you wanted to see how many rpms you have:
Example using `wc` command:

```
[root@localhost Pacakges]# ls -ltr | wc -l
```

- Run an `ls -ltr` command and pipe it with a `wc -l` command to output the count of the number of lines that the `ls` command throws.

Output:

```
3973
```

- We are gonna copy all of these packages to our local drive, the local folder that we just created.

First make sure that you have enough space in your Linux machine
Example using `du` command:

```
[root@localhost Pacakges]# du -sh .
```

- This line will give you the disk space of the current location.

Output:

```
3.7G
```

- Check also with `df -h` to see that we have enough space available in the `/` directory.

Copy the contents of the `Packages/`
Example using `cp` command:

```
[root@localhost Pacakges]# cp -rv .* /localrepo/
```

- The `-r` option stands for "recursive" and specifies to copy directories recursively.
- The `-v` option stands for "verbose" and specifies to explain what is being done.
- Remember the `.` is referring to the relative path but you can also run `cp -rv /run/media/mmarin/RHEL-9-4-0-BaseOS-x86_64/Packages/*` with the same effect.
- It will copy every single `rpm` to the "localrepo" directory.
- To see progress on copying all of the rpms you can travel to the location directory and repeatedly run `ls -l | wc -l` to count the lines of the output of `ls -l`.
- Use `*` because you are copying everything under `Packages/`
- Note that the location or the place that these packages will be copy to is our previously created directory `/localrepo/`.

Tell Linux to instead of going outside it can go locally to this repository

- There are files in `/etc/yum.repos.d/`
  - These are all the files that tells Linux to go to these different mirrors

Remove everything in this directory
Example using `rm` command:

```
[root@localhost yum.repos.d]# rm -rf /etc/yum.repos.d/*
```

Create a new repo with vi
Example using `vi` command:

```
[root@localhost yum.repos.d]# vi local.repo
```

- Note we are creating this file inside the current location which is the `/etc/yum.repos.d/` directory.
- Add the following values:

File editor:

```
[centos7]
name=centos7
baseurl=file:///localrepo/
enabled=1
gpgcheck=0
~
~
~
~
:
```

- These are some of the default values you have to put into that file.
- Note the `/localrepo/` is the directory that we previously created.

- remember to save the file with `:wq!`

The `createrepo` command

- For CentOS 8/9 = command not found, but you can install it with `dnf install createrepo`
- This command is available in RHEL 9

Example using `createrepo` command:

```
[root@localhost yum.repos.d]# createrepo /localrepo/
```

- Remember the `/localrepo/` is the directory we previously created for this purpose.
- `createrepo` will create the database that will be available to our `yum`/`dnf` command

Output:

```
Spawning worker 0 with 3971 pkgs
...
```

- It will make these packages available to my local machine as a repository
- It can take up to 3-8 minutes.

Clean all cache from the old repository
Example using `yum` command:

```
[root@localhost yum.repos.d]# yum clean all
```

- This line is going to clear out any cache from the old repository.

Example using `yum` command:

```
[root@localhost yum.repos.d]# yum repolist all
```

- This line should tell you your full list of repositories.

Now if you try to install any packages through the `yum`/`dnf` command the Linux machine is gonna go into the repository and check if it has the package that matches the one it is looking for. Note the system is not going out to the internet or to your Red Hat satellite, the system is actually going to your local repository which is sitting within your Linux machine.