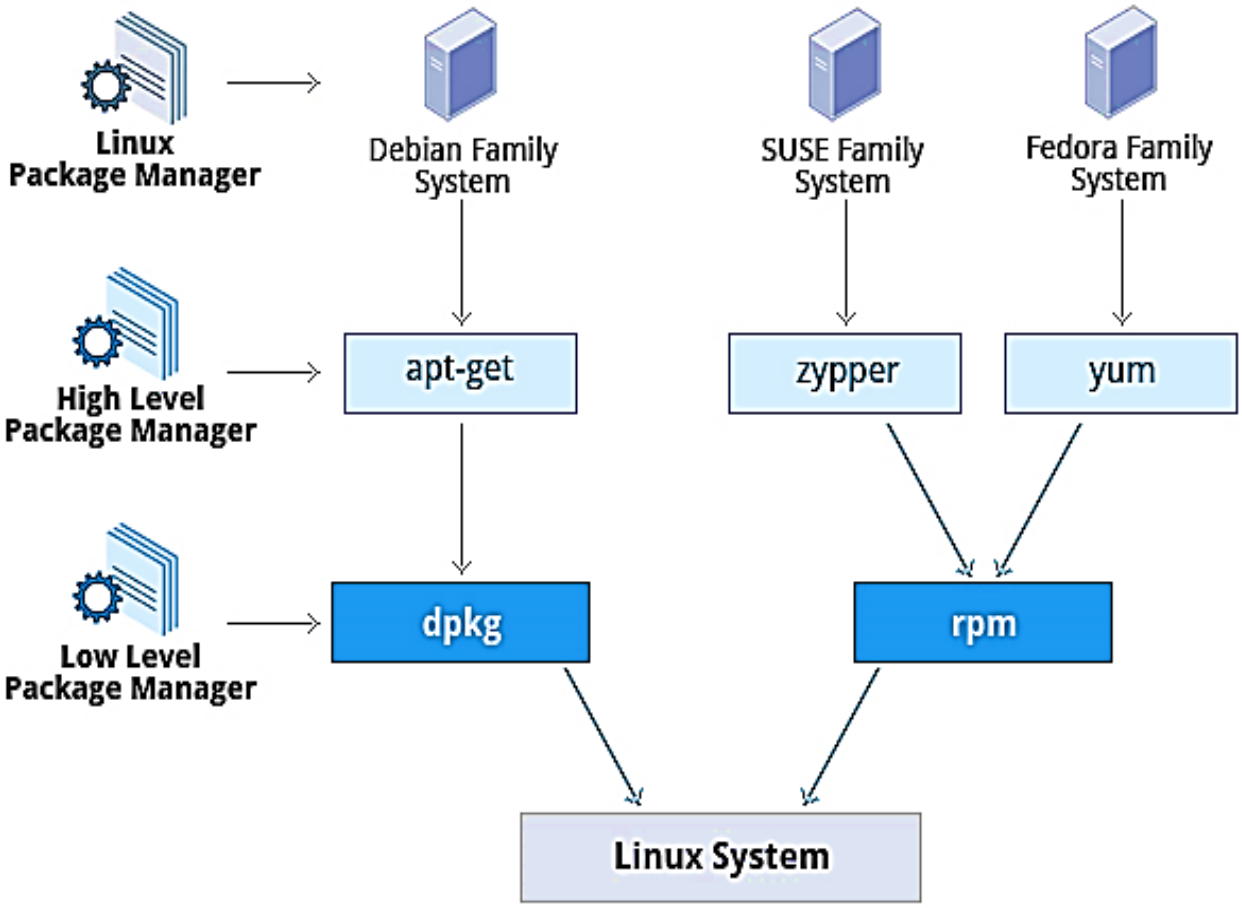
Advance Linux Administration Marathon

Day 5

Topics:-

1. Package Management
2. RPM Package



Managing software

- Installing, patching, and removing software packages on Linux machines is one of the common tasks every sysadmin has to do.
- Package management is a method of installing, updating, removing, and keeping track of software updates from specific repositories (repos) in the Linux system.

There are two types of package management:

1. deb
2. rpm

Redhat based distros use following method to install software packages:

1) RPM (Redhat Package Manager)

2) YUM (Yellowdog Updater Manager) / DNF (Dandified YUM)

Redhat Package Manager (RPM)

- RPM is a popular package management tool in Red Hat Enterprise Linux-based distribution.
- Using RPM, you can install, uninstall, and query individual software packages.
- Still, it cannot manage dependency resolution like YUM.
- RPM does provide you useful output, including a list of required packages.
- An RPM package consists of an archive of files and metadata.

Key Features of RPM:

• **Low-level package manager**: It works directly with rpm files.

• **Package integrity checking**: It can check the integrity and authenticity of packages.

• **Dependency management**: RPM identifies package dependencies but doesn't resolve them on its own (YUM/DNF resolves dependencies).

RPM Naming Conventions (Nomenclature)

The name of an RPM package typically follows this format:

<package_name>-<version><release>.<architecture>.rpm

package_name: The name of the package.

version: The software version.

release: The release number of the package Build.

architecture: The architecture(e.g., x86_64, noarch).

For example:

httpd-2.4.37-30.e18.x86_64.rpm

- **Httpd** : Package name.
- **2.4.37**: Version
- **30.e18**: Release (30th release for RHEL 8).
- **x86_64**: Architecture.

rpm commands

- **To install a rpm package**

  Syntax: rpm -ivh <package name>

  Example: rpm -ivh vsftpd

- **To query a command packages**

  Syntax: rpm -qf <path of command>

  Example: rpm -qf /usr/bin/ping

- **To see documentation of packages**

  Syntax: rpm -qd <path of command>

  Example: rpm -qd /usr/sbin/useradd

- **To list all the files installed by a package**

  Syntax: rpm -ql <Package name>

  Example: rpm -ql httpd

- **To upgrade .rpm package**

  Syntax: rpm -U <package file>

  Example: rpm -U perl

- **To erase package**

  Syntax: rpm -e <package name>

  Example: rpm -e tree

Options with RPM Commands:

•**-v:** Provides verbose output.

•**-h:** Displays a progress bar (used with install/upgrade).

•**-nodeps:** Ignores dependency checks (not Recommendseded.)

•**-test:** Tests the installation/upgrade without making changes

YUM (Yellowdog Updater, Modified)

- Package management tool for installing, updating, removing, and managing software packages in Red Hat Enterprise Linux.
- Performs dependency resolution when installing, updating, and removing software packages.
- All the repos are at /etc/yum.repos.d. Directory.
- Repository is a storage location from which your system retrieves and installs OS updates and applications.
- Each repository is a collection of software hosted on a remote server and intended to be used for installing and updating software packages on Linux systems.
- Red Hat Enterprise Linux 9 is distributed through two main repositories:
    1. BaseOs 2.AppStream

How YUM Works:

Dependency Resolution: YUM automatically resolves package dependencies by fetching necessary packages from configured repositories.

Repository Management: YUM connects to online or Local repositories, where software packges and

Medata are stored.

How to configure yum client?

Syntax: vi /etc/yum.repos.d/test.repo

```
[myrepo 1]
name=any
baseurl=file:///mnt/BaseOS
enabled=1
gpgcheck=0
[myrepo 2]
name=any1
baseurl=file:///mnt/AppStream
enabled=1
gpgcheck=0
```

YUM commands

yum install :- Installs the specified packages

remove :- Removed the specified package

search :- Searches package metadata for keywords

update :- Updates each packet to the latest version

repolist :- List repositories

info:- List description

Difference between RPM and YUM

RPM	YUM
RPM stands for Redhat Package Manager.	YUM stands for Yellow Dog Updater.
RPM can install only single package at a time.Ex:- rpm install gedit	YUM can install multiple package at a time. Ex:- yum install httpd vsftpd
RPM cannot resolve the dependencies.	YUM can resolve dependencies automatically.
Cannot rollback with RPM	YUM can rollback any changes.

What is Subscription Manager?

Subscription Manager (subscription-manager) is a tool used in RHEL to manage system entitlements and subscriptions to Red Hat's software channels (RHSM - Red Hat Subscription Management).

Why Use RHSM?

- **Access to Official Repositories**: It grants access to official Red Hat repositories.
- **Automatic Updates**: Ensures the system receives updates and security patches.
- **Entitlement Management**: Ensures you're using valid Red Hat subscriptions in compliance with licensing agreements.

Common Subscription Manager Commands:

Subscription-manager :- Register the system register : - With Red Hat.

Subscription-manager :- Unregister the system unregister

Step to Build Your Own RPM Package

To build your own RPM package in RHEL 9, you'll need to follow several steps, from setting up the required environment to packaging your application. Here's a step-by-step guide to building and using your RPM package in RHEL 9

1. Install Required Tools: Before begin, ensure you have the necessary tools for building RPM packages:

```
[root@localhost ~]# yum install rpm-build rpmdevtools
Updating Subscription Management repositories.
Extra Packages for Enterprise Linux 9 - x86_64                  836  B/s | 5.4 kB     00:06
Extra Packages for Enterprise Linux 9 - x86_64                  524 kB/s |  23 MB     00:44
Red Hat Enterprise Linux 9 for x86_64 - AppStream (RPMs)        501  B/s | 4.5 kB     00:09
Red Hat Enterprise Linux 9 for x86_64 - AppStream (RPMs)        1.8 MB/s |  41 MB     00:22
Red Hat Enterprise Linux 9 for x86_64 - BaseOS (RPMs)           593  B/s | 4.1 kB     00:07
Package rpm-build-4.16.1.3-29.el9.x86_64 is already installed.
Package rpmdevtools-9.5-1.el9.noarch is already installed.
Dependencies resolved.
Nothing to do.
Complete!
```

2. Set up RPM Build Environment: you need to create a directory structure for building the RPM package:

```
[root@localhost ~]# rpmdev-setuptree
```

This will create the following directory structure

Your home directory:

```
[root@localhost ~]# ls rpmbuild
BUILD  RPMS  SOURCES  SPECS  SRPMS
[root@localhost ~]# tree rpmbuild/
rpmbuild/
├── BUILD
├── RPMS
├── SOURCES
├── SPECS
└── SRPMS

5 directories, 0 files
[root@localhost ~]#
```

3. Create a shell Script: First, create a simple shell script that prints “Hello Harshal”

```
[root@localhost ~]# vim hello.sh
[root@localhost ~]# cat hello.sh
#!/bin/bash
echo "Hello Harshal.."
echo "rpm test file success."
```

4. Move the Script to the SOURCES Directory

Move your shell script to the SOURCES directory:

```
[root@localhost ~]# cp hello.sh ~/rpmbuild/SOURCES/
[root@localhost ~]# ls rpmbuild/SOURCES/
hello.sh
[root@localhost ~]#
```

Create a SPEC File Now, create a SPEC File in the SPEC directory:

```
[root@localhost SOURCES]# vim hello.spce
[root@localhost SOURCES]# cat hello.spce
Name: hello_Harshal
Version: 1.0
Release: 1%{?dist}
Summary: A simple script to print Hello Hatshal

License: GPLv2
URL: http://example.com
Source: hello.sh

%description
This package contains a simple shell script that prints "Hello Vaibhav".

%prep
%setup -q
%build
%install
rm -rf %{buildroot}
install -D -m 755 hello.sh %{buildroot}/usr/local/bin/hello_Harshal

%changelog
* Sat Oct 5 2024 Harshal Pawar <harshal.email@example.com> - 1.0-1
- Initial package creation
[root@localhost SOURCES]#
```

```
[root@localhost ~]# tree rpmbuild/
rpmbuild/
├── BUILD
├── RPMS
├── SOURCES
│   ├── hello.sh
│   └── hello.spce
├── SPECS
└── SRPMS

5 directories, 2 files
```

Key RPM Packaging Concepts

1. **Source0**: Specifies the source tarball containing the application code.
2. **BuildRequires**: Lists dependencies needed to build the package.
3. **Requires**: Defines runtime dependencies required for the software to function.
4. **%prep**: Prepares the build environment, such as extracting the source.
5. **%build**: Contains instructions for compiling the software.
6. **%install**: Specifies how to install files into the build root.
7. **%files**: Lists all files to be included in the final RPM package.

Thank you..!