**Package Management**

1. **Red Hat Package manager**
2. **Yellow Dog Update modified**
3. **Red Hat Package manager**

**Packages and Packaging**

A **software package** is a set of files organized in a directory structure and makes up a Red Hat software application. Files contained in a package include installable scripts, configuration tiles, commands and associated documentation. The documentation provides detailed instructions on how to install and uninstall the package, manual pages of the files and commands included and any other specific information pertaining to installation and usage.

All data related to packages is stored at a central location and includes information such as package versioning, location it is installed at and checksum values. This allows package management tools to efficiently handle package administration tasks by referencing this data.

**Package Naming Convention**

Red Hat software packages follow a standard naming convention. Typically, there are four parts to it. The first part contains the package name, the second includes the version of it, the third includes the package release (revision or build) and the last part tells the processor architecture the package is built for. An installable package name always has the .rpm extension. The extension is removed after the package has been installed. For example:

**sendmail-8.13.8-2.el5.i386.rpm (package name before it is installed)**

**sendmail-8.l3.8-2.el5 (package name after it has been installed)**

Here is a description of each part in the package name:

- sendmail - package name

- 8.13.8 - package version

- 2 - package release

- el5 - stands for Enterprise Linux 5.

- i386 - processor architecture the package is built for. If you see "noarch" instead the package will be platform-independent and can be installed on any hardware architecture. If you see "src", it will contain source code for the package.

**Package Dependency**

A package to be loaded may require the presence of certain files or other packages in order for a successful installation of it. Similarly, many software packages require certain files or other packages to be present in order for them to be able to run and operate smoothly. This is referred to as**package dependency** where software depends on other software for a successful installation or run.

**Package Database**

Metadata information of installed package files is stored in the */var/lib/rpm* directory. This directory location is referred to as**package database**. This database is referenced by package management tools to obtain information such as ownership, permissions, timestamp and size of files. This database also contains information about package dependencies. The information contained in here help package management commands verify dependencies and file attributes, upgrade and uninstall an existing package, and add new packages.

**Package Repository**

A **package *repository*** is a storage location from where one or several software packages can be downloaded at cost or cost-free for installation. Red Hat maintains its own repositories for this purpose. In addition to accessing many other Internet-based repositories, you can create your own and add packages to it for later installation on one or more systems.

**Package Administration Tools**

Several commands are available to perform package management functions. The table below provides a list along with a short description of each.

|  |  |
| --- | --- |
| **Command** | **Description** |
| *rpm* | Installs, removes, updates and verifies a package, this command does not automatically satisfy package dependencies. |
| *rpm2cpio* | Extracts packages. |
| *rpmbuild* | Builds new RPMs based on customization. |
| *rpmquery* | Queries on packages. |
| *rpmverify* | Verifies package attributes. |
| *yum* | Front-end to the rpm command. Installs, removes and updates packages from one or more repositories and automatically satisfies dependencies. Also allows to search and list package information. |
| *pup* | Installs and updates packages. |
| *pirut* | Graphical tool to install and update packages. |

**Managing Packages with rpm**

This section discusses package management tasks including listing, installing, upgrading, freshening, querying, removing, extracting, validating and verifying packages using the *rpm* command.

First let’s take a look at the options provided by the **rpm** command:

|  |  |
| --- | --- |
| **Option** | **Description** |
| -a (-all) | Displays all packages. |
| -c (--configfiles) | Displays configuration files. |
| -d (--docfiles) | Displays documentation files. |
| -e (--erase) | Removes a package. |
| --f (--file) | Displays information about the specified file. |
| -F (--freshen) | Upgrades an existing package. An older version of the package must exist in order to upgrade it. |
| --force | Installs a package even if the same version already exists. |
| -h (--hash) | Displays progress of package installation / upgrade. |
| -i (--info) | Displays basic information about a package. |

|  |  |
| --- | --- |
| **Option** | **Description** |
| -i (--install) | Installs a package. |
| -K (--checksig) | Validates the signature and also the package integrity. |
| -I (--list) | Lists files in a package. |
| -p (--package) | Verifies an installed package against an installable package. |
| -q (--query) | Queries and displays packages. You can use the *rpmquery* command instead. |
| --replacepkgs | Overwrites existing packages. |
| -R (--requires) | Lists dependencies without which a package cannot be installed. |
| -U (--upgrade) | Upgrades an existing package or installs if not already installed. |
| -v or --vv | Displays detailed information. |
| -V (--verify) | Verifies the integrity of package files. You can use the *rpmverifv* command instead. |

**Command examples:**

-listing installed packages:

**# rpm -qa | grep mail**

sendmail-cf-8.13.8-8.el5

mailcap-2.1.23-1.fc6

procmail-3.22-17.1.el5.centos

mailx-8.1.1-44.2.2

launchmail-4.0.0-2.el5

sendmail-8.13.8-8.el5

fetchmail-6.3.6-1.1.el5\_3.1

**# rpmquery -a | grep mail**

sendmail-cf-8.13.8-8.el5

mailcap-2.1.23-1.fc6

procmail-3.22-17.1.el5.centos

mailx-8.1.1-44.2.2

launchmail-4.0.0-2.el5

sendmail-8.13.8-8.el5

fetchmail-6.3.6-1.1.el5\_3.1

These commands list an updated list of all packages currently loaded on the system, with **pipe grep** used to select only the mail packages.

**Homework: Install MySQL-server first with rpm method, then uninstall and use yum for the installation.**

1. **Yellow Dog Update modified**

The yum command (yellow dog updater modified) is a text-based utility used for package management. This tool requires that your system has access to one or more configured software repositories such as RHN with a valid user account. Alternatively, packages to be installed may be downloaded and stored in a local yum repository. The location of the repository is then defined in the /etc/yum.repos.d directory. The primary benefit of using this tool is that it performs dependency checks by itself and downloads any required packages automatically in order to successfully install the specified package. With multiple repositories defined, yum can extract the specified software package from wherever it finds it. The default yum repository is RHN. When the yum command is executed the first time on a system to connect to the CentOS repository, it downloads header information associated with software packages and keeps them in cache. The next time you access CentOS repo via yum, it will download only the updated headers into cache.

Before getting into details, let us take a look at the table below, which provides a list of options commonly used with the yum command.

|  |  |
| --- | --- |
| **Option** | **Description** |
| check-upate | Checks if any updates are available for the installed packages. |
| clean | Synchronizes package header information. |
| groupinstall | Installs or updates a group of packages. |
| info | Displays package header information. |
| install | Installs the specified package(s) or updates them if already installed. |
| list | Lists packages that are installed or available for installation or update. |
| localinstall | Installs or updates packages located locally on the system |
| remove | Removes the specified package |
| search | Searches for packages that contain the specified string |
| update | Updates packages if already installed |

**YUM Configuration File**

The key configuration file for yum is /etc/yum.conf . The default contents are listed below:

**# cat /etc/yum.conf**

[main]

cachedir=/var/cache/yum

keepcache=0

debuglevel=2

logfile=/var/log/yum.log

distroverpkg=redhat-release

tolerant=1

exactarch=1

obsoletes=1

gpgcheck=1

plugins=1

# Note: yum-RHN-plugin doesn't honor this.

metadata\_expire=1h

# Default.

# installonly\_limit = 3

# PUT YOUR REPOS HERE OR IN separate files named file.repo

# in /etc/yum.repos.d

The table explains various directives in the file:

|  |  |
| --- | --- |
| **Directive** | **Description** |
| cachedir | Specifies the location to store *yum* downloads. Default is */var/cache/vuni.* |
| keepcache | Specifies whether to store the cache of packages and headers following a successful installation. Default is 0 (disabled). |
| debuglevel | Specifies the level at which the debug is to be recorded in the Iogfile. Default is 2. |
| iogfile | Specifies the location of the log file for *yum* activities. Default is  */var/log/vum.log.* |
| pkgpolicy | Specifies the version to be downloaded/installed. Default is newest. |
| distroverpkg | Specifies where to get the version. Default is redhat-release, which obtains the information from */etc/redhat-release* file. |
| tolerant | Specifies whether to ignore minor errors. Default is 1 (enabled). |
| exactarch | Specifies the CPU architecture for the package to be downloaded. Default is 1 (enabled). |
| obsoletes | Checks and removes any obsolete packages. Default is 1 (enabled). |
| gpgcheck | Specifies whether to check the GPG signature for package authenticity. Default is 1 (enabled). |
| plugins | Specifies to include plug-ins with the packages to be downloaded. Default is 1 (enabled). |
| metadata\_expire | Specifies a lifetime for header data downloaded from RUN. Default is 1 hour. The *yum* command automatically downloads the latest package header information if *yum* is not used for this long. This is to ensure that header data is up to date. |

Some command examples:

**# yum install postfix**

Loaded plugins: fastestmirror

Loading mirror speeds from cached hostfile

\* addons: lnx.apollo-hw.ro

\* base: lnx.apollo-hw.ro

\* extras: lnx.apollo-hw.ro

\* updates: lnx.apollo-hw.ro

Setting up Install Process

Resolving Dependencies

--> Running transaction check

---> Package postfix.x86\_64 2:2.3.3-2.1.el5\_2 set to be updated

--> Finished Dependency Resolution

Dependencies Resolved

…………………………………………………….

**# yum remove postfix**

Loaded plugins: fastestmirror

Setting up Remove Process

Resolving Dependencies

--> Running transaction check

---> Package postfix.x86\_64 2:2.3.3-2.1.el5\_2 set to be erased

--> Finished Dependency Resolution

Dependencies Resolved

…………………………………………………….