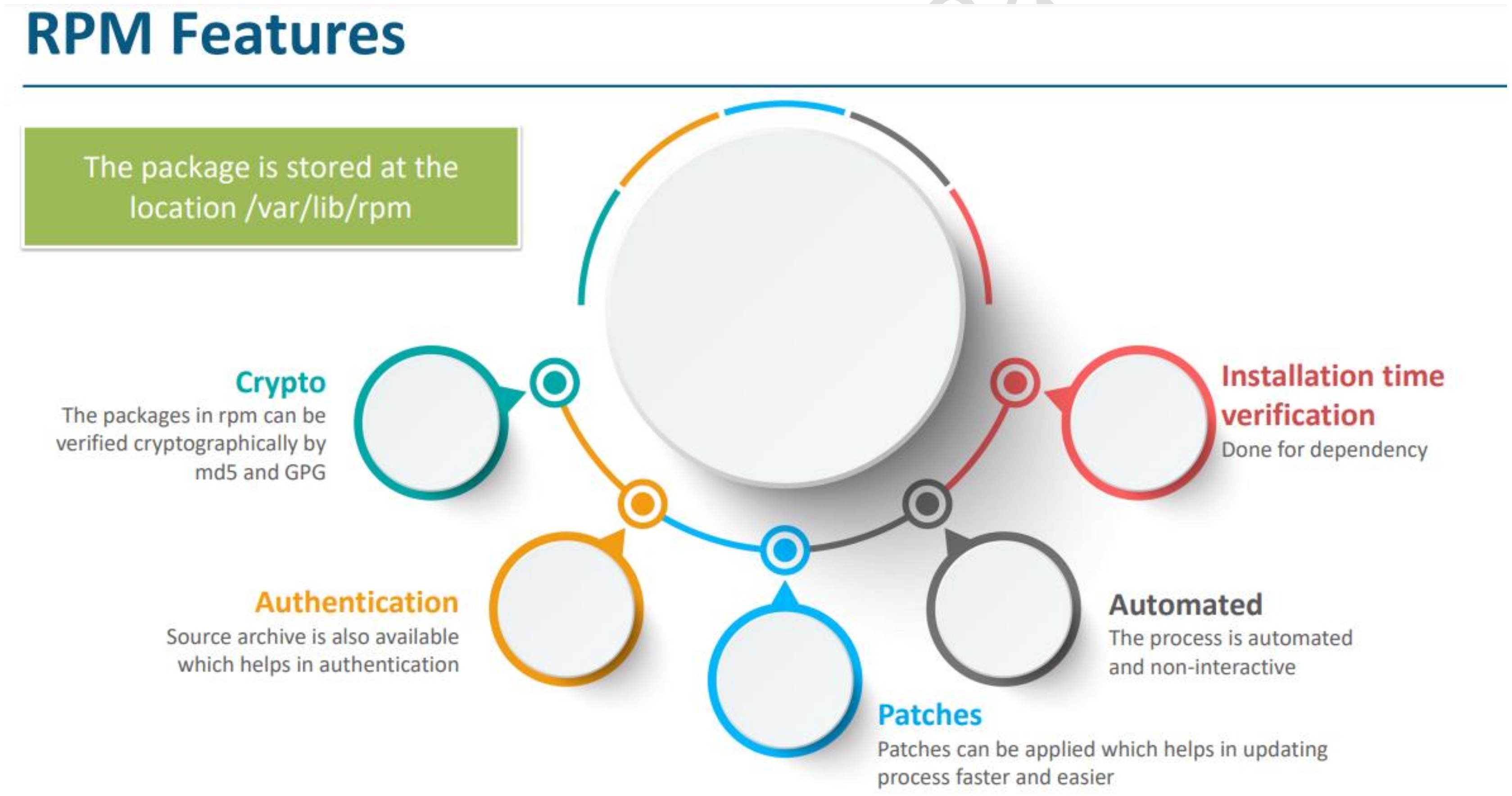
Installing and Updating Software Packages

RPM SOFTWARE PACKAGES

RPM stands for REDHAT PACKAGE MANAGEMENT

And which is developed by Red hat itself to manage software's.


RPM Features
The package is stored at the location /var/lib/rpm
Crypto
The packages in rpm can be verified cryptographically by md5 and GPG
Authentication
Source archive is also available which helps in authentication
Patches
Patches can be applied which helps in updating process faster and easier
Automated
The process is automated and non-interactive
Installation time verification
Done for dependency

And always it consist of

name-version-release.architecture.rpm

Example: **apacheds-2.0.0.AM25-x86_64.rpm**
Name-version.Release arch .rpm

RPM is a default open source and most popular package management utility for RedHat based systems like (RHEL, CentOS and Fedora)

- **The tool allows users to install, update, uninstall, query, verifies and manages system software packages in Unix/Linux operating systems.**
- **The RPM formerly known as .rpm file, which includes compiled software programs and libraries needed by the packages.**
- **This utility only works with packages that built on .rpm format.**
  RPM keeps the information of all the installed packages under /var/lib/rpm database.

#rpm -qf /etc/yum.repos.d (filenames)

#rpm -q yum (version of package)

#rpm -ql yum (list files installed by package)

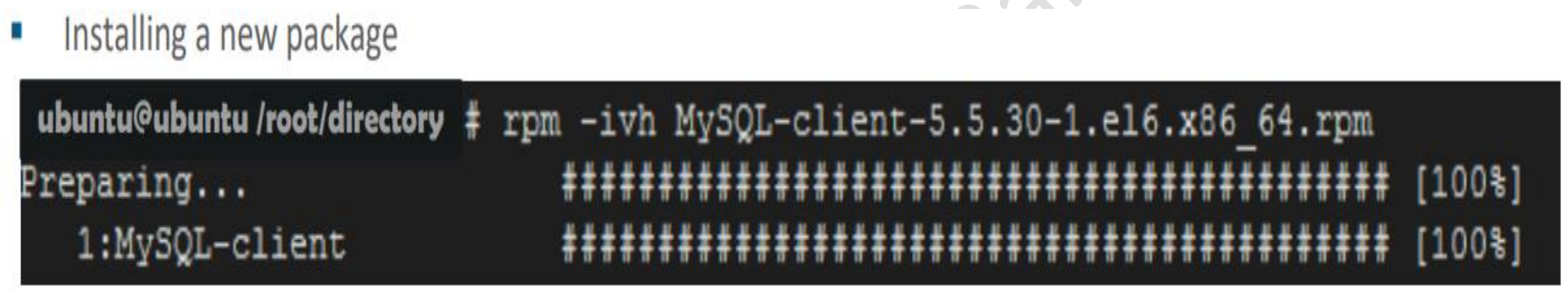
#rpm -qi yum (full details of packages)

#rpm -qc openssh-clients (list conf file installed by package)

#rpm -qd openssh-clients (doc installed by packages)

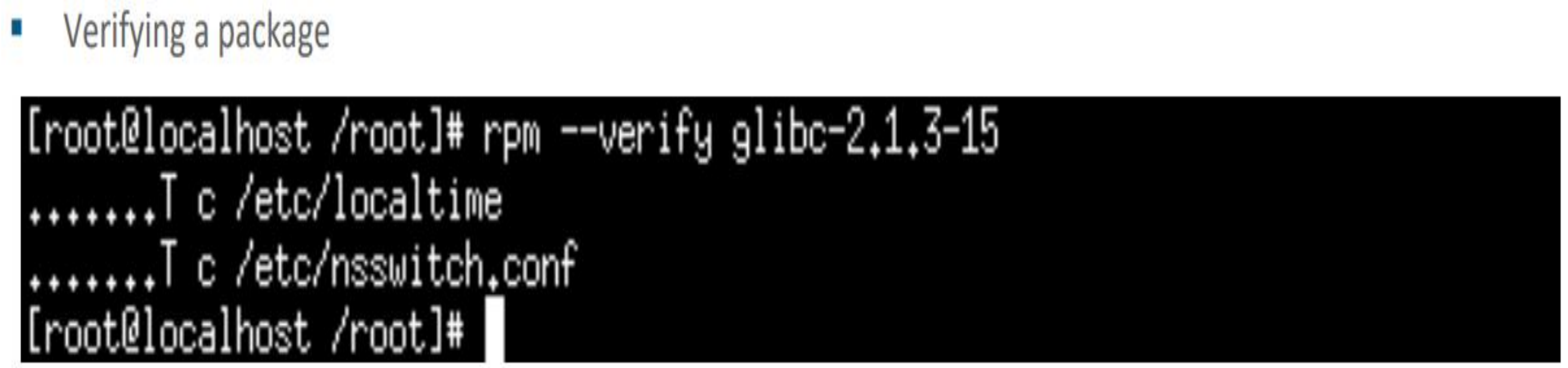
#rpm -q --scripts openssh-server (List shell scripts that run before or after the package is installed or removed)

#rpm -ivh wonderwidgets-1.0-4.x86_64.rpm



- Installing a new package

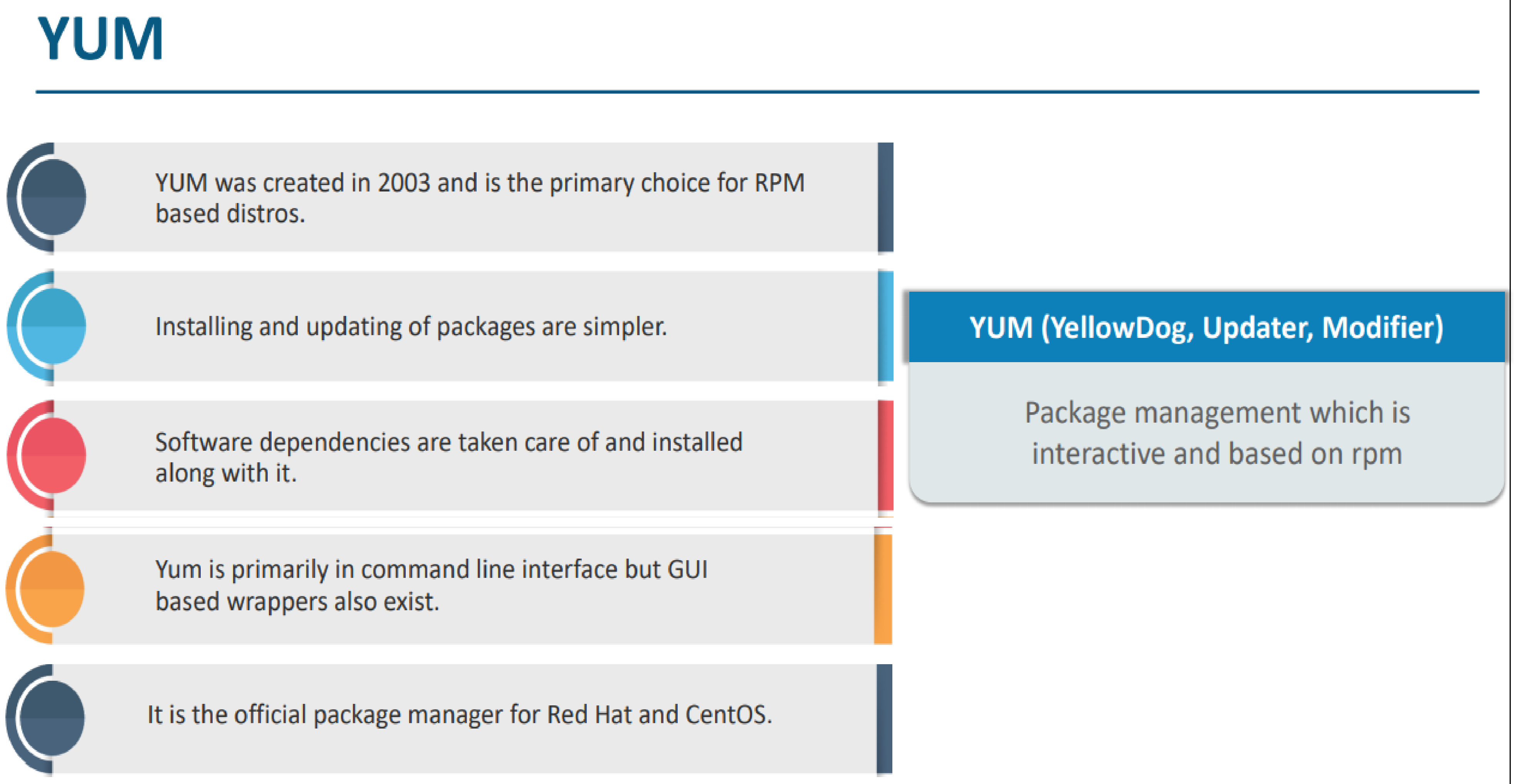
```
ubuntu@ubuntu /root/directory # rpm -ivh MySQL-client-5.5.30-1.el6.x86_64.rpm
Preparing...                ########################################### [100%]
   1:MySQL-client           ########################################### [100%]
```



- Verifying a package

```
[root@localhost /root]# rpm --verify glibc-2.1.3-15
.......T c /etc/localtime
.......T c /etc/nsswitch.conf
[root@localhost /root]#
```

YUM



YUM

- YUM was created in 2003 and is the primary choice for RPM based distros.
- Installing and updating of packages are simpler.
- Software dependencies are taken care of and installed along with it.
- Yum is primarily in command line interface but GUI based wrappers also exist.
- It is the official package manager for Red Hat and CentOS.

YUM (YellowDog, Updater, Modifier)

Package management which is interactive and based on rpm

YUM stands for **Yellowdog Updater Modified**

It allow you to install, update remove and get info about software or their dependency

➔ It is an open source command-line as well as graphical based package management tool for RPM (RedHat Package Manager) based Linux systems.

➔ YUM uses numerous third party repositories to install packages automatically by resolving their dependencies issues.

➔ It is located in /etc/yum.repos.d/ directory

➔ It have .repo extension, to be recognized by YUM

- **#yum help**
- **List**
- **Search all ‘web’**
- **Info httpd**
- **Provides pathname**
- **Install httpd**
- **Update**
- **Remove httpd**
- All install and remove transactions are logged in ***/var/log/dnf.rpm.log.***
- **Yum history**
- **Yum history undo 3**

RPM and YUM

RPM is the package manager which installs the package while YUM is repository management tool which will fetch the package

YUM fetches the appropriate package as required by our Linux

RPM can alert you to dependencies but is unable to source additional packages. Yum finds out the dependencies and also source the additional packages

YUM by default has multiple repository location in its configuration file to fetch package for installation

➔ yum also has the concept of groups, which are collections of related software installed together for a particular purpose

Here we have two type of groups

- **Regular group**: collection of packages
- **Environment Group**: collection of regular groups

The packages or groups provided by a group may be

- **mandatory** (they must be installed if the group is installed),
- **Default** (normally installed if the group is installed)
- **Optional** (not installed when the group is installed, unless specifically requested).

Yum group list

Yum group info "RPM Development Tools"

ENABLING YUM SOFTWARE REPOSITORIES

As per the subscription we can avail repository

Yum repolist all

Yum-config-manager

--enable

--disable

If we want to enable any third party repo

We can create a repo in **/etc/yum.repos.d/**

It must have .repo extension

It CONSIST OF

[repoid]

Name=opt

Baseurl=url or ftp or local file

Gpgcheck=as a key {enable or disable}

Enabled=1/0

Or else we can use

Yum-config-manager –add-repo=”url /……/..”

To enable EPEL8(Extra Packages for Enterprise Linux)

```
[user@host ~]$ rpm --import [illegible]
```

```
[user@host ~]$ yum install http://dl.fedoraproject.org/pub/epel/8/x86_64/e/epelrelease-8-
2.noarch.rpm
```

MANAGING PACKAGE MODULE STREAMS

Traditionally for managing different version of application and its related we are managing different **repos**

RHEL 8 include module stream in repository

To control must imp versions of packages, and this tech is called **Modularity,** *it allow a single repository to host multiple version of app and its dependency.*

In RHEL 8

We can find two divisions

BaseOS: it provides the core part of OS as rpm packages

& AppStream: it includes necessary part of the system and a wide range of app and its streams, as part of RHSC (Redhat software collections**)**

<u>**MODULE:**</u> set of rpm packages that are consistent set of belong together

Each module can have multiple **MODULE STREAMS & PROFILES**

MODULE STREAMS:

It holds multiple diff versions of content

PROFILES:

A list of certain packages to be install together for particular use-case

#**yum module list**

yum module list perl

# yum module info perl

# yum module install -y perl

```
# yum module remove -y perl

# yum module disable perl

# yum module install perl:5.24
```

dpkg

01 Dpkg is the main package management system in Debian and similar OSes

02 It is used to install, build, remove, and manage packages

03 The package for it has an extension of .deb at the end

04 Dpkg is a low level tool and APT is the commonly used high level tool as it can deal with complex tasks involved in package management

05 The dpkg database is located under /var/lib/dpkg

Install Package

Use command ‘-i’ to install a package

Syntax

```
dpkg –i <package name>
```

Example : # dpkg –i python2.7.deb

Syntax

```
# dpkg –s python
```

To check if a package is installed or not use ‘s’ option.

List Package

- Use command '-l' to list a package with dpkg.

Syntax

```
dpkg –l <package_name>
```

Example : # dpkg –l python

To list all packages, don't add a package name.

```
# dpkg –l
```

To view content of a package, use '-c' option.

```
# dpkg –c python2.7.deb
```

Remove Package

- To remove a package we must use package name and not the original one with .deb extension.

Syntax

```
dpkg –r <package name>
```

```
Example : # dpkg –r python
```

apt-get

Apt-get is the command line interface to handle package using APT library.

It is the default package management system for Debian-like distro like Ubuntu.

It is an efficient way of handling packages in your system.

Dependencies are managed automatically.

Upgrades and removal are handled carefully to maintain the stability of the system.

It has an external GUI support with tools like synaptic, aptitude, etc.

apt-cache

Apt-cache is the command line interface to search apt software packages.

This tool is used to search software packages and get information about them.

One can search for a package without having exact name of the package.

The data is fetched from different sources listed in sources.list file.

/var/cache/apt/archives/ contains already downloaded packages to avoid downloading them again if one needs to re-install a package after removing it.

List & Search Package

- Use command 'pkgnames' to list packages starting with a particular string.

Syntax

```
apt-cache pkgnames <package_name>
```

Example : # apt-cache pkgnames python

Syntax

```
# apt-cache search python
```

Use command 'search' to search for a package with a particular name.

Check Package Information

- Use command 'show' to get details about a package.

Syntax

```
apt-cache show <package_name>
```

Example : # apt-cache show python

Syntax

```
# apt-cache showpkg python
```

To check dependencies of a package use 'showpkg' option.

Update Package

- Use command 'update' to update a package.

Syntax

```
apt-get update <package_name>
```

Example : # apt-get update python

To update the whole system, don't provide package name.

```
# apt-get update
```

To install a package but prevent from upgrading if already installed use '- -no-upgrade' option.

```
# apt-get install python - -no-upgrade
```

Install Package

- Use command 'install' to install a package.

Syntax

```
apt-get install <package_name>
```

Example : # apt-get install python

To install multiple packages together, provide multiple package name after install.

```
# apt-get install python mysql
```

To install multiple package having a particular string, use wildcard.

```
# apt-get install '*name'
```

Remove Package

- Use command ‘remove’ to remove a particular package.

Syntax

```
apt-get remove <package_name>
```

Example : # apt-get remove python

Syntax

```
# apt-get remove - -purge python
```

Removing a package doesn’t remove its configuration file. To remove configuration files along with it, append with ‘purge’ option.