

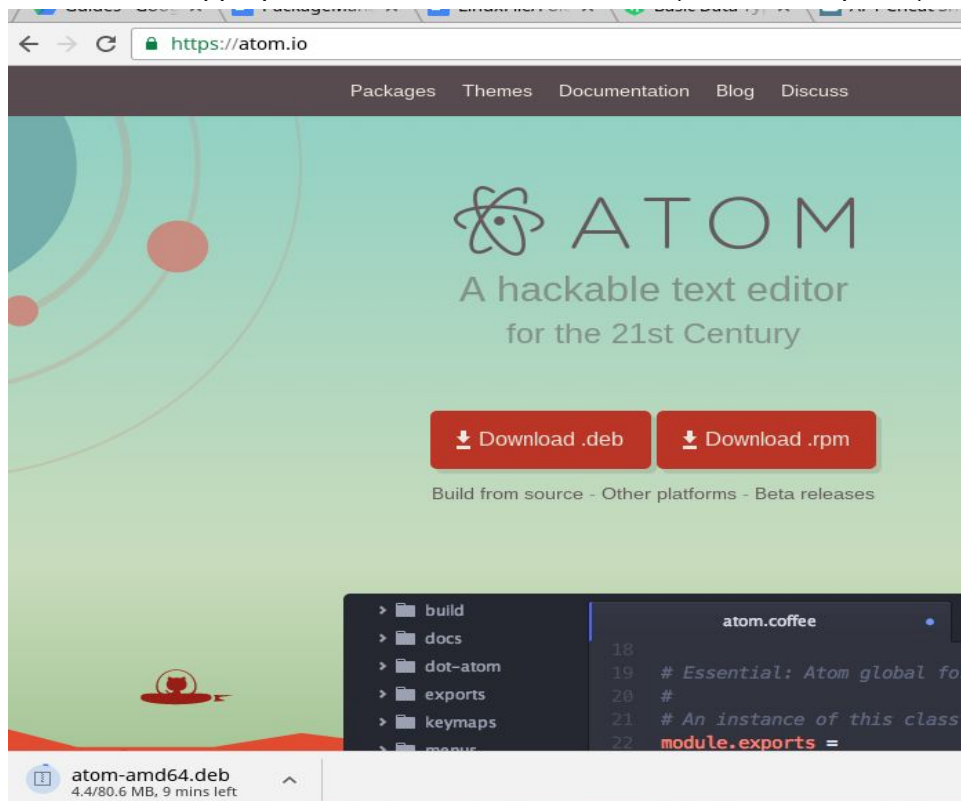
4.0 Package Managers

Package managers are a way of installing, removing, upgrading software on a Linux system. There are a few very popular package managers that exist, and just how every flavor of Linux has its own unique way of doing things, the package manager is not an exception. For this chapter I will be going over the package managers for Debian and Redhat based Distros as well as some other information I find suitable for this topic. Eventually I will go over, PACMAN and PORTAGE (Arch and Gentoo) pk managers. For this chapter I'm going to assume you already have a **redhat** based distro installed somewhere.

4.1 Debian Package Management

Debian has a few tools that work towards the overall way Debian handles its packages. This first tool is DPKG. DPKG package provides the low-level infrastructure for handling the installation and removal of Debian software packages. In other words, DPKG is the lowest possible level of handling packages through a Debian system. Let's look at how simple this tool is.

Let's install a .deb package, through the command line. This is equivalent to Microsoft's .exe or .msi. Navigate to this website. <https://atom.io/> and download the .deb . make sure you download the appropriate *CPU architecture* as well (Refer to chapter 1).



To install a .deb package, use the command like so `dpkg -i debFileName.deb` . In our case we have the atom package.

```
lu@awesome ~/Downloads $ ls
atom-amd64.deb
lu@awesome ~/Downloads $ sudo dpkg -i atom-amd64.deb
[sudo] password for lu:
Selecting previously unselected package atom.
(Reading database ... 199243 files and directories currently installed.)
Preparing to unpack atom-amd64.deb ...
Unpacking atom (1.12.1) ...
dpkg: dependency problems prevent configuration of atom:
 atom depends on git; however:
  Package git is not installed.
dpkg: error processing package atom (--install):
 dependency problems - leaving unconfigured
Processing triggers for gnome-menus (3.13.3-6ubuntu3) ...
Processing triggers for desktop-file-utils (0.22-1ubuntu5) ...
Processing triggers for mime-support (3.59ubuntu1) ...
Errors were encountered while processing:
 atom
lu@awesome ~/Downloads $
```

If you get the following error, that is okay, there is a **dependency** problem. A dependency can be a package, library or framework that needs to be installed in order for the software package you need to run and install properly. Let's fix this by running

```
lu@awesome ~/Download $ sudo apt-get install git
```

This is where the other tools like APT and aptitude come in handy, if we were to install the atom package via these tools. APT and aptitude would automatically download and install the dependencies for us. Pretty neat right?!

Still getting errors?? Use Aptitude!

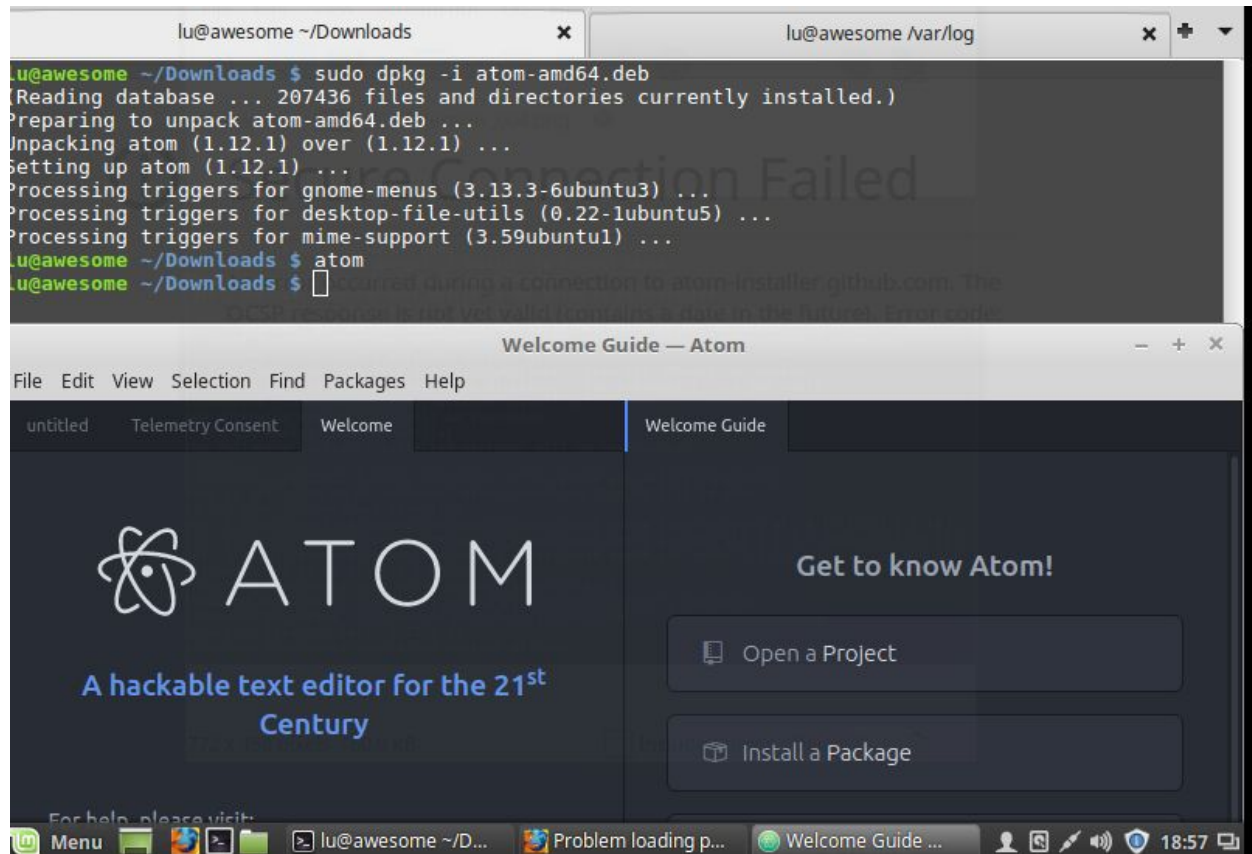
```
lu@awesome ~/Download $ sudo aptitude install git
```

```
lu@awesome ~/Downloads $ sudo apt-get -f install git
Reading package lists... Done
Building dependency tree
Reading state information... Done
You might want to run 'apt-get -f install' to correct these:
The following packages have unmet dependencies:
 git : Depends: liberror-perl but it is not going to be installed
       Depends: git-man (> 1:2.7.4) but it is not going to be installed
       Depends: git-man (< 1:2.7.4..) but it is not going to be installed
E: Unmet dependencies. Try 'apt-get -f install' with no packages (or specify a solution).
lu@awesome ~/Downloads $ aptitude install git
E: Could not open lock file /var/lib/dpkg/lock - open (13: Permission denied)
E: Unable to lock the administration directory (/var/lib/dpkg/), are you root?
lu@awesome ~/Downloads $ sudo aptitude install git
The following NEW packages will be installed:
 git git-man{a} liberror-perl{a}
The following partially installed packages will be configured:
 atom
0 packages upgraded, 3 newly installed, 0 to remove and 441 not upgraded.
Need to get 3,760 kB of archives. After unpacking 25.6 MB will be used.
Do you want to continue? [Y/n/?] y
get: 1 http://archive.ubuntu.com/ubuntu xenial/main amd64 liberror-perl all 0.17-1.2 [19.6 kB]
get: 2 http://archive.ubuntu.com/ubuntu xenial/main amd64 git-man all 1:2.7.4-0ubuntu1 [735 kB]
get: 3 http://archive.ubuntu.com/ubuntu xenial/main amd64 git amd64 1:2.7.4-0ubuntu1 [3,006 kB]
Fetched 3,760 kB in 53s (70.5 kB/s)
Selecting previously unselected package liberror-perl.
(Reading database ... 206622 files and directories currently installed.)
Preparing to unpack .../liberror-perl_0.17-1.2_all.deb ...
Unpacking liberror-perl (0.17-1.2) ...
Selecting previously unselected package git-man.
```

Now hopefully, if you encountered various errors like I did, you should finally be able to install this package

```
lu@awesome ~/Download $ sudo dpkg -i atom-amd64.deb
```

```
lu@awesome ~/Download $ atom
```



Why would we need to install .deb packages if APT and Aptitude can do this for us? Well there are a lot of third party software that are not available in the official debian repositories , and a .deb file makes it easier than downloading and compiling a package from source. Essentially a .deb package is a special archive that extracts and installs into your system.

Here are some more fun things you can do with dpkg that are very helpful when you are trying to find which package are installed on your system.

The list of installed packages can be obtained with (This will display all the packages that are installed on the system.

```
lu@awesome ~/Download $ sudo dpkg -l
```



```
lu@awesome ~/Downloads $ sudo dpkg -l
Desired=Unknown/Install/Remove/Purge/Hold
| Status=Not/Inst/Conf-files/Unpacked/halF-inst/trig-aWait/Trig-pend
|/ Err?=(none)/Reinst-required (Status,Err: uppercase=bad)
||/ Name                 Version                Architecture           Description
++-+-----+-----+-----+-----+
ii accountsservice      0.6.40-2ubuntu        amd64                  query and manipulate user account informat
ii acl                   2.2.52-3               amd64                  Access control list utilities
ii acpi-support          0.142                  amd64                  scripts for handling many ACPI events
ii acpid                 1:2.0.26-1ubun        amd64                  Advanced Configuration and Power Interface
ii add-apt-key           1.0-0.5                all                    Command line tool to add GPG keys to the A
ii adduser               3.113+nmu3ubun        all                    add and remove users and groups
ii adwaita-icon-theme    3.18.0-2ubuntu        all                    default icon theme of GNOME (small subset)
ii adwaita-icon-theme    3.18.0-2ubuntu        all                    default icon theme of GNOME
ii alsa-base             1.0.25+dfsg-0u        all                    ALSA driver configuration files
ii alsa-utils            1.1.0-0ubuntu5        amd64                  Utilities for configuring and using ALSA
ii anacron               2.3-23                amd64                  cron-like program that doesn't go by time
ii apg                   2.2.3.dfsg.1-2        amd64                  Automated Password Generator - Standalone
ii app-install-data      15.10                  all                    Ubuntu applications (data files)
ii apt                   1.2.10ubuntu1         amd64                  commandline package manager
ii apt-clone             0.4.1ubuntu1          all                    Script to create state bundles
ii apt-transport-http    1.2.10ubuntu1         amd64                  https download transport for APT
ii apt-utils             1.2.10ubuntu1         amd64                  package management related utility program
ii apt-xapian-index      0.47ubuntu8           all                    maintenance and search tools for a Xapian
ii aptdaemon             1.1.1+bzr982-0        all                    transaction based package management servi
ii aptdaemon-data        1.1.1+bzr982-0        all                    data files for clients
ii aptitude              0.7.4-2ubuntu2        amd64                  terminal-based package manager
ii aptitude-common       0.7.4-2ubuntu2        all                    architecture independent files for the apt
ii apturl                0.5.2+linuxmin        all                    install packages using the apt protocol -
```

All of the packages will be displayed into a list. Too long of a list right? What if we want to filter these results? Or output them into a file?

To filter these results to a very specific string you are trying to find, try this command.

```
lu@awesome ~/Downloads $ sudo dpkg --get-selections |grep apache
```

This will display which packages have the string “atom” installed.

```
lu@awesome ~/Downloads $ sudo dpkg --get-selections |grep atom
atom                                install
libatomic1:amd64                  install
lu@awesome ~/Downloads $
```

What if we are on the run, and want the output saved into a file? This can be done by directing the output into a file.

```
lu@awesome ~/Downloads $ sudo dpkg -l > output.txt
```

```
lu@awesome ~/Downloads $ sudo dpkg -l > output.txt
lu@awesome ~/Downloads $ ls
atom-amd64.deb  output.txt
lu@awesome ~/Downloads $ nano output.txt
lu@awesome ~/Downloads $
```

What we just did with the `>` argument is we directed the standard output into a text file. If you view the file, you can see that we basically shoved the `sudo dpkg -f` command output into the file. Read the file and see that it would be the same as if it were displayed in the terminal.

Before we move on into the APT manager, let's talk a little about the standard outputs of the linux system.

Lastly, if you want to remove a package simply use the `--remove` or `-r` argument

lu@awesome ~/Download \$ `sudo dpkg -r apache`

```
lu@awesome ~/Downloads $ sudo dpkg --remove apache
dpkg: warning: ignoring request to remove apache which isn't installed
lu@awesome ~/Downloads $
```

APT is the Advanced Package Tool, an advanced interface to the Debian packaging system which provides the `apt-get` program. It provides command line tools for searching and managing packages, and for querying information about them, as well as low-level access to all features of the `libapt-pkg` library. For more information, see the User's Guide in `/usr/share/doc/apt-doc/guide.html/index.html` (you will have to install the `apt-doc` package).

Starting with Debian Jessie, some frequently used `apt-get` and `apt-cache` commands have an equivalent via the new `apt` binary. This means some popular commands like `apt-get update`, `apt-get install`, `apt-get remove`, `apt-cache search`, or `apt-cache show` now can also be called simply via `apt`, say `apt update`, `apt install`, `apt remove`, `apt search`, or `apt show`. The following is an overview of the old and new commands:

apt-get update	-> apt update
apt-get upgrade	-> apt upgrade
apt-get dist-upgrade	-> apt full-upgrade
apt-get install package	-> apt install package
apt-get remove package	-> apt remove package
apt-get autoremove	-> apt autoremove
apt-cache search string	-> apt search string
apt-cache policy package	-> apt list -a package
apt-cache show package	-> apt show package
apt-cache showpkg package	-> apt show -a package

The `apt` tool merges functionality of `apt-get` and `apt-cache` and by default has a fancier colored output format, making it more pleasant for humans. For usage in scripts or advanced use cases, `apt-get` is still preferable or needed.

apt-get provides a simple way to retrieve and install packages from multiple sources using the command line. Unlike dpkg, apt-get does not understand .deb files, it works with the packages proper name and can only install .deb archives from a source specified in /etc/apt/sources.list. apt-get will call dpkg directly after downloading the .deb archives[4] from the configured sources.

Some common ways to use apt-get are:

To update the list of packages known by your system, you can run:

```
lu@awesome ~/Download $ sudo apt update
```

(you should execute this regularly to update your package lists)

To install the foo package and all its dependencies, run:

```
lu@awesome ~/Download $ sudo apt install foo
```

```
apt install foo
```

To remove the foo package from your system, run:

```
lu@awesome ~/Download $ sudo apt remove foo
```

To remove the foo package and its configuration files from your system, run:

```
lu@awesome ~/Download $ sudo apt purge foo
```

To upgrade all the packages on your system (without installing extra packages or removing packages), run:

```
lu@awesome ~/Download $ sudo apt list --upgradable
```

To upgrade all the packages on your system, and, if needed for a package upgrade, installing extra packages or removing packages, run:

```
lu@awesome ~/Download $ sudo apt upgrade
```

(The command upgrade keeps a package at its installed obsolete version if upgrading would need an extra package to be installed, for a new dependency to be satisfied. The full-upgrade command is less conservative.)

```
lu@awesome ~/Download $ sudo apt full-upgrade
```

Note that you must be logged in as root to perform any commands that modify packages.

Note that apt-get now also installs recommended packages as default, and thanks to its robustness it's the preferred program for package management from console to perform system installation and major system upgrades.

The apt tool suite also includes the apt-cache tool to query the package lists. You can use it to find packages providing specific functionality through simple text or regular expression queries and through queries of dependencies in the package management system. Some common ways to use apt-cache are:

To find packages whose description contain word:

```
lu@awesome ~/Download $ sudo apt search KEYWORD
```

To print the detailed information of a package:

```
lu@awesome ~/Download $ sudo apt show PACKAGE
```

To print the packages a given package depends on:

```
lu@awesome ~/Download $ sudo apt-cache depends PACKAGE
```

To print detailed information on the versions available for a package and the packages that reverse-depends on it:

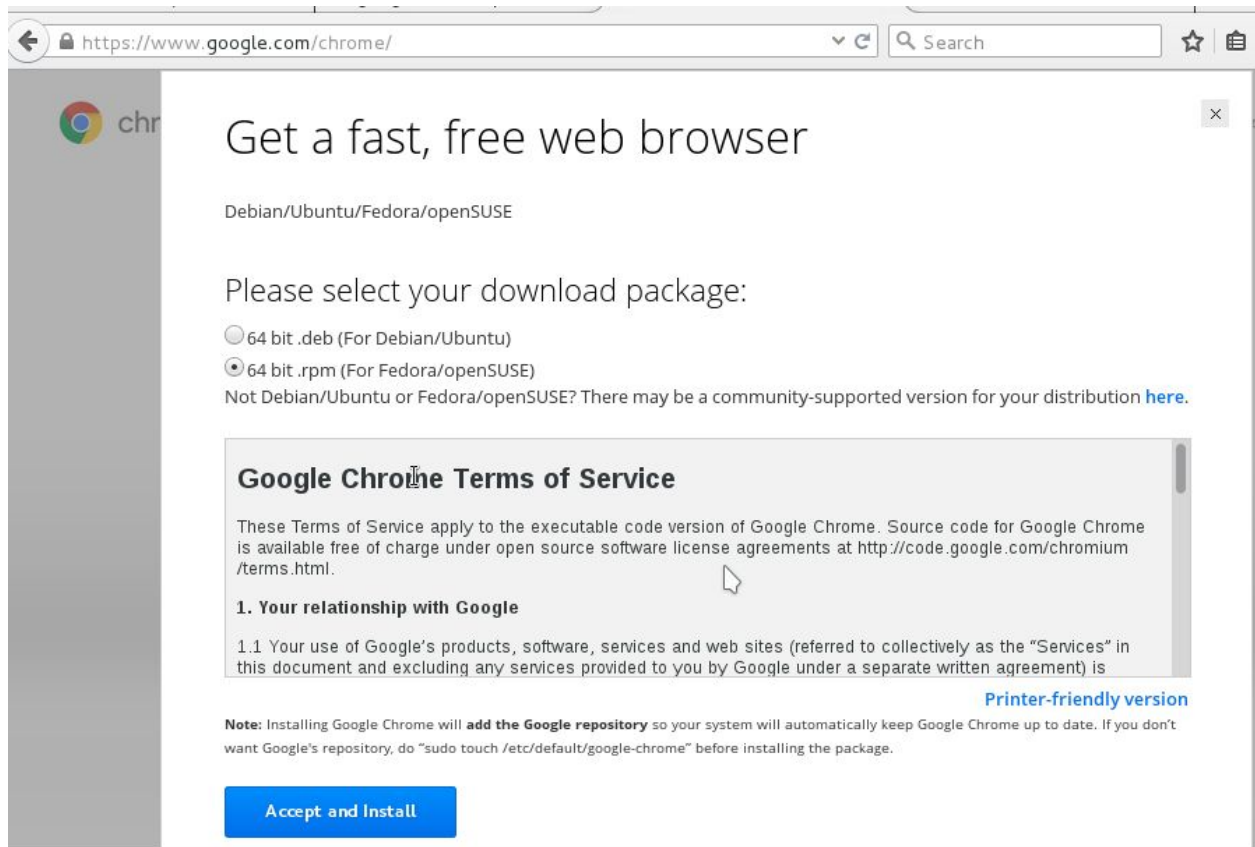
```
lu@awesome ~/Download $ sudo apt-cache showpkg PACKAGE
```

That's it for managing packages with Debian. At least for this competition you will not have to do anything crazy or special. So the basics of these tools will take you far.

4.2 Red Hat Package Management

First off, just like how Debian has a low level, package manager (**DPKG**), Red Hat also has a low level package manager called **RPM**. **That manages .rpm files, equivalent to .deb**. Let's talk about this first, there have been times where **YUM** (that's equivalent to APT) is broken and everything gets REAL, quick. So sometimes we just have to download some **.RPM** files and install them. So let's get to it.

Inside your CentOS virtual machine navigate open up firefox and navigate to the google chrome website and download the .rpm file. Lets install that baby. Since google chrome is not in the official repositories off CentOS.



Navigate to the dir and execute the following command. Switch to root and run
`[root@localhost Downloads] rpm -ivh google-chrome-stable_current_x86_64.rpm`
-i to install , v to print verbose (-V to verify) and h to print the has.

```
File Edit View Search Terminal Help
[lu@localhost Downloads]$ su root
Password:
ABRT has detected 1 problem(s). For more info run: abrt-cli list
[root@localhost Downloads]# ls
google-chrome-stable_current_x86_64.rpm
[root@localhost Downloads]# man rpm
[root@localhost Downloads]# rpm -ivh google-chrome-stable_current_x86_64.rpm
warning: google-chrome-stable_current_x86_64.rpm: Header V4 DSA/SHA1 Signature,
key ID 7fac5991: NOKEY
error: Failed dependencies:
    lsb >= 4.0 is needed by google-chrome-stable-54.0.2840.100-1.x86_64
    libXss.so.1()(64bit) is needed by google-chrome-stable-54.0.2840.100-1.x
86_64
[root@localhost Downloads]#
```

Now we will get dependencies errors, so let's install those.

`[root@localhost Downloads] yum install lsb libxss.so.1`


```
lu@localhost:/home/lu/Downloads
File Edit View Search Terminal Help
redhat-lsb-core          x86_64 4.1-27.el7.centos.1 base 38 k
redhat-lsb-cxx           x86_64 4.1-27.el7.centos.1 base 16 k
redhat-lsb-desktop       x86_64 4.1-27.el7.centos.1 base 20 k
redhat-lsb-languages     x86_64 4.1-27.el7.centos.1 base 18 k
redhat-lsb-printing      x86_64 4.1-27.el7.centos.1 base 16 k
redhat-lsb-submod-multimedia x86_64 4.1-27.el7.centos.1 base 15 k
redhat-lsb-submod-security x86_64 4.1-27.el7.centos.1 base 15 k
spax                     x86_64 1.5.2-13.el7 base 260 k
systemtap-sdt-devel      x86_64 2.8-10.el7 base 65 k
Updating for dependencies:
glibc                    x86_64 2.17-106.el7_2.8 updates 3.6 M
glibc-common             x86_64 2.17-106.el7_2.8 updates 11 M
glibc-devel              x86_64 2.17-106.el7_2.8 updates 1.0 M
glibc-headers            x86_64 2.17-106.el7_2.8 updates 663 k
nss-softokn-freebl       x86_64 3.16.2.3-14.2.el7_2 updates 204 k

Transaction Summary
=====
Install 2 Packages (+82 Dependent packages)
Upgrade ( 5 Dependent packages)

Total size: 49 M
Total download size: 32 M
Is this ok [y/d/N]: 
```

Now let's talk about **YUM**, Yum is the **Redhat** package manager. YUM can query for information about available packages, fetch packages from repositories, install and uninstall them, and update an entire system to the latest available version. Yum performs automatic dependency resolution when updating, installing, or removing packages, and thus is able to automatically determine, fetch, and install all available dependent packages. YUM is equivalent to debian's **APT or Aptitude**. In this case we need the **lsb package and libxss.so.1 library** for our googel chrome .rpm and those are availble in the offial repo. once we ran the **[root@localhost Downloads] yum install lsb libxss.so.1** command , YUM automatically is going to resolve the dependencies that RPM could not do for us. This make YUM very powerful. So type **y** and hit **enter**. And hit y if any licence things come up.

Lets try it again

```
[root@localhost Downloads] rpm -ivh google-chrome-stable_current_x86_64.rpm
```

```
File Edit View Search Terminal Help
[root@localhost Downloads]# rpm -ivh google-chrome-stable_current_x86_64.rpm
warning: google-chrome-stable_current_x86_64.rpm: Header V4 DSA/SHA1 Signature, key ID 7fac5991: NOKEY
error: Failed dependencies:
    libXss.so.1()(64bit) is needed by google-chrome-stable-54.0.2840.100-1.x86_64
[root@localhost Downloads]# 
```

Now we still have that dependency problem! Why? We did install lsb and libxss.so.1 right? WRONG! Well we installed the lsb package but YUM had a hard time installing the libxss.so.1 file.



Here is where we will kind of use YUM to debug this annoying dependency problem. Lets search for the package to see if it is even there. Lets run multiple queries to see where this *library* file is hiding(**reference chapter 2 about .so files**). We know it's a library and not a package because of the .so extension.

```
[root@localhost Downloads] yum search libXss.so.1
```

NOTHING will be found. Lets try trimming the search query down

```
[root@localhost Downloads] yum search libXss.so
```

Nothing will be found again. Maybe yum does not like the DOT ? " . "

```
[root@localhost Downloads] yum search libXss
```

Now We have a match! This is what we are looking for. The X.Org X11 libXss runtime library is what we need in order for the google chrome package to install properly. The libXss.so1 library is most likely an inner dependency for the **libXScrnSaver.x86_64** package and will be pulled in once we install it. So install it!

```

File Edit View Search Terminal Help
Loaded plugins: fastestmirror, langpacks
Loading mirror speeds from cached hostfile
* base: centos.mirror.ndchost.com
* extras: mirror.supremebytes.com
* updates: mirrors.usc.edu
Warning: No matches found for: libXss.so.1
No matches found
[root@localhost Downloads]# yum search libXss.so
Loaded plugins: fastestmirror, langpacks
Loading mirror speeds from cached hostfile
* base: centos.mirror.ndchost.com
* extras: mirror.supremebytes.com
* updates: mirrors.usc.edu
Warning: No matches found for: libXss.so
No matches found
[root@localhost Downloads]# yum search libXss.so
Loaded plugins: fastestmirror, langpacks
Loading mirror speeds from cached hostfile
* base: centos.mirror.ndchost.com
* extras: mirror.supremebytes.com
* updates: mirrors.usc.edu
Warning: No matches found for: libXss.so
No matches found
[root@localhost Downloads]# yum search libXss
Loaded plugins: fastestmirror, langpacks
Loading mirror speeds from cached hostfile
* base: centos.mirror.ndchost.com
* extras: mirror.supremebytes.com
* updates: mirrors.usc.edu
===== N/S matched: libXss =====
libXScrnSaver.i686 : X.Org X11 libXss runtime library
libXScrnSaver.x86_64 : X.Org X11 libXss runtime library

Name and summary matches only, use "search all" for everything.
[root@localhost Downloads]#

```

[root@localhost Downloads] yum install libXScrnSaver.x86_64

Note there are two of the same package, that is CPU architecture dependant, install the appropriate one for the type of system you are running (**Reference Chapter 1**).

```

Dependencies Resolved
=====
Package Arch Version Repository Size
=====
Installing:
libXScrnSaver x86_64 1.2.2-6.1.el7 base 24 k
Transaction Summary
=====
Install 1 Package

Total download size: 24 k
Installed size: 40 k
Is this ok [y/d/N]: y
Downloading packages:
libXScrnSaver-1.2.2-6.1.el7.x86_64.rpm | 24 kB 00:00:00
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
  Installing : libXScrnSaver-1.2.2-6.1.el7.x86_64 1/1
  Verifying : libXScrnSaver-1.2.2-6.1.el7.x86_64 1/1

Installed:
libXScrnSaver.x86_64 0:1.2.2-6.1.el7

Complete!
[root@localhost Downloads]#

```

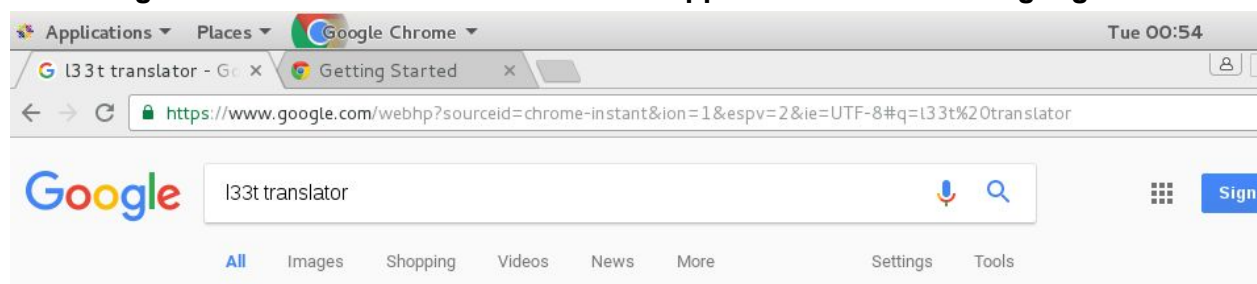
Now Chrome should behave and install properly. Try it one last time

```
[root@localhost Downloads] rpm -ivh google-chrome-stable_current_x86_64.rpm
```

```
[root@localhost Downloads]# rpm -ivh google-chrome-stable_current_x86_64.rpm
warning: google-chrome-stable_current_x86_64.rpm: Header V4 DSA/SHA1 Signature, key ID 3541c874: no signature found
Preparing... ##### [100%]
Updating / installing...
 1:google-chrome-stable-54.0.2840.10##### [100%]
Redirecting to /bin/systemctl start atd.service
[root@localhost Downloads]# google-chrome
```

BOOMSHAKALAKA!

Now Google chrome should be available in the Applications->internet-> google chrome



This following section will be of rpm commands that you should try out on your own. These are a couple of commands that I think are relevant to the competition. For more info about RPM please read https://www.centos.org/docs/5/html/Deployment_Guide-en-US/s1-rpm-using.html

Uninstall a package. Use the -e flag to uninstall an rpm. Note i will be using the google chrome rpm as an example. -e for empty

```
[root@localhost Downloads] rpm -e google-chrome-stable
```

Note I am using the name of the package that is already installed. Not the actual package name, where the application came from (google-chrome-stable_future_x86_64.rpm)

Updating a package and replace it with a newer version. -U for update

```
[root@localhost Downloads] rpm -Uvh google-chrome-stable_future_x86_64.rpm
```

Verify an installed package against and RPM package file

```
[root@localhost Downloads] rpm -Vp google-chrome-stable_future_x86_64.rpm
```

Verify all the packages in the system

```
[root@localhost Downloads] rpm -Va
```

Query a package to find more information about it

```
[root@localhost Downloads] rpm -q google-chrome-stable
```



```
[root@localhost Downloads]# rpm -q google-chrome-stable
google-chrome-stable-54.0.2840.100-1.x86_64
```

That's it for RPM!. Let's move on to YUM. we already know how to install a package and we already know how to search for packages. Lets run a couple more examples.

```
[root@localhost /] yum install emacs
```

```
Installed:
  emacs.x86_64 1:24.3-18.el7
,
Dependency Installed:
  ImageMagick.x86_64 0:6.7.8.9-15.el7_2  OpenEXR-libs.x86_64 0:1.7.1-7.el7      emacs-common.x86_64 1:24.3-18.el7
  ilmbase.x86_64 0:1.0.3-7.el7          libXaw.x86_64 0:1.0.12-5.el7      libblockfile.x86_64 0:1.08-17.el7
  libotf.x86_64 0:0.9.13-4.el7          libwmf-lite.x86_64 0:0.2.8.4-41.el7_1
Complete!
[root@localhost /]#
```

```
[root@localhost /] yum remove google-chrome-stable
```

```
[root@localhost /]# yum remove google-chrome-stable
Loaded plugins: fastestmirror, langpacks
Resolving Dependencies
--> Running transaction check
--> Package google-chrome-stable.x86_64 0:54.0.2840.100-1 will be erased
--> Finished Dependency Resolution
```

Dependencies Resolved

Package	Arch	Version	Repository	Size
Removing:				
google-chrome-stable	x86_64	54.0.2840.100-1	installed	170 M

Transaction Summary

Remove 1 Package

```
[root@localhost /] yum search cheese
```

```
===== N/S matched: cheese =====
cheese-libs-devel.i686 : Development files for cheese-libs
cheese-libs-devel.x86_64 : Development files for cheese-libs
cheese.x86_64 : Application for taking pictures and movies from a webcam
cheese-camera-service.x86_64 : Webcam D-Bus service
cheese-libs.i686 : Webcam display and capture widgets
cheese-libs.x86_64 : Webcam display and capture widgets

Name and summary matches only, use "search all" for everything.
[lu@localhost /]$
```

Lets check to see which packages need to be updated on our system with *yum check-update*

```
[root@localhost /] yum check-update
```


This could be a really long list, so we are gonna pipe it to `less`. This will allow us to scroll through the output. After you are done scrolling hit “`:q`” to exit out.

```
[root@localhost /] yum check-update | less
```

Now to really update our entire system and all of the dependencies

```
[root@localhost /] yum update
```

```
Transaction Summary
=====
Install      2 Packages (+1 Dependent package)
Upgrade    227 Packages

Total size: 387 M
Is this ok [y/d/N]:
```

Update an individual package

```
[root@localhost /] yum update google-chrome-stable
```

Yum Allows us to update security packages without all the other packages in the system. We can do this with the `--security` argument or the `update-minimal --security` arguments.

```
[root@localhost /] yum update --security
```

```
[root@localhost /] yum update-minimal --security
```

For example, assume that:

the **kernel-3.10.0-1** package is installed on your system;

the **kernel-3.10.0-2** package was released as a security update;

the **kernel-3.10.0-3** package was released as a bug fix update.

Then `yum update-minimal --security updates` the package to **kernel-3.10.0-2**,
and `yum update --security` updates the package to **kernel-3.10.0-3**.

Now let's list all the packages available in the repo's

```
[root@localhost /] yum list all
```

This is going to be a very long list, are you starting to see a pattern? Every time we have a long output we can always save the output to a file, then search the inner contents of that file, or we can grep the output for specific string. This is basically what `yum search` does, except `yum search` gives us more info.

```
[root@localhost /] yum list all |grep httpd
```

```
[root@localhost /] yum search httpd
```

```
[root@localhost /]# yum list all |grep httpd
httpd.x86_64                                2.4.6-40.el7.centos.4      updates
httpd-devel.x86_64                        2.4.6-40.el7.centos.4      updates
httpd-manual.noarch                       2.4.6-40.el7.centos.4      updates
httpd-tools.x86_64                        2.4.6-40.el7.centos.4      updates
libmicrohttpd.i686                        0.9.33-2.el7               base
libmicrohttpd.x86_64                     0.9.33-2.el7               base
libmicrohttpd-devel.i686                  0.9.33-2.el7               base
libmicrohttpd-devel.x86_64                 0.9.33-2.el7               base
libmicrohttpd-doc.noarch                   0.9.33-2.el7               base
[root@localhost /]# yum search httpd
Loaded plugins: fastestmirror, langpacks
Loading mirror speeds from cached hostfile
 * base: centos.mirror.ndchost.com
 * extras: mirror.supremebytes.com
 * updates: mirrors.usc.edu
===== N/S matched: httpd =====
libmicrohttpd-devel.i686 : Development files for libmicrohttpd
libmicrohttpd-devel.x86_64 : Development files for libmicrohttpd
libmicrohttpd-doc.noarch : Documentation for libmicrohttpd
httpd.x86_64 : Apache HTTP Server
httpd-devel.x86_64 : Development interfaces for the Apache HTTP server
httpd-manual.noarch : Documentation for the Apache HTTP server
httpd-tools.x86_64 : Tools for use with the Apache HTTP Server
libmicrohttpd.i686 : Lightweight library for embedding a webserver in applications
libmicrohttpd.x86_64 : Lightweight library for embedding a webserver in applications
mod_auth_mellon.x86_64 : A SAML 2.0 authentication module for the Apache Httpd Server
mod_dav_svn.x86_64 : Apache httpd module for Subversion server
```

Now lets try to search for a specific query and try to find out if anything related to that query is installed in our system. Not only will this command search for installed packages but it will search for the packages that are available in the repo via the @anaconda flag. Note the * character tells linux “everything” in this case yum list installed google(include everything that has google in the system and repo). In this case we have google chrome installed and some fonts available in the repo.

[root@localhost /] yum list installed google*

```
[root@localhost /]# yum list installed google*
Loaded plugins: fastestmirror, langpacks
Loading mirror speeds from cached hostfile
 * base: centos.mirror.ndchost.com
 * extras: mirror.supremebytes.com
 * updates: mirrors.usc.edu
Installed Packages
google-chrome-stable.x86_64                54.0.2840.100-1            installed
google-crosextra-caladea-fonts.noarch      1.002-0.4.20130214.el7     @anaconda
google-crosextra-carlito-fonts.noarch      1.103-0.2.20130920.el7     @anaconda
```

Use these commands to find out information about which repo's are being used.

[root@localhost /] yum list repolist -v

[root@localhost /] yum repoinfo

[root@localhost /] yum repolist all

Lets check packages for more info

[root@localhost /] yum info empathy

```
Installed Packages
Name      : empathy
Arch      : x86_64
Version   : 3.12.10
Release   : 2.el7
Size      : 13 M
Repo      : installed
From repo : anaconda
Summary   : Instant Messaging Client for GNOME
URL       : http://live.gnome.org/Empathy
License   : GPLv2+
Description : Empathy is powerful multi-protocol instant messaging client which
            : supports Jabber, GTalk, MSN, IRC, Salut, and other protocols.
            : It is built on top of the Telepathy framework.
```

Now the last few things we will be talking about is groups, transaction history, and mirror list. Groups allows us to install packages that are grouped together, such as different desktops and different IDE's

[root@localhost /] yum groups summary

```
[root@localhost /]# yum groups summary
Loaded plugins: fastestmirror, langpacks
There is no installed groups file.
Maybe run: yum groups mark convert (see man yum)
Loading mirror speeds from cached hostfile
 * base: centos.mirror.ndhost.com
 * extras: mirror.supremebytes.com
 * updates: mirrors.usc.edu
Available Environment Groups: 10
Available Groups: 10
Done
```

[root@localhost /] yum list ids

```
[root@localhost ~]# yum group list ids
Loaded plugins: fastestmirror, langpacks
There is no installed groups file.
Maybe run: yum groups mark convert (see man yum)
Loading mirror speeds from cached hostfile
* base: centos.mirror.ndchost.com
* extras: mirror.supremebytes.com
* updates: mirrors.usc.edu
Available Environment Groups:
Minimal Install (minimal)
Compute Node (compute-node-environment)
Infrastructure Server (infrastructure-server-environment)
File and Print Server (file-print-server-environment)
Basic Web Server (web-server-environment)
Virtualization Host (virtualization-host-environment)
Server with GUI (graphical-server-environment)
GNOME Desktop (gnome-desktop-environment)
KDE Plasma Workspaces (kde-desktop-environment)
Development and Creative Workstation (developer-workstation-environment)
Available Groups:
Compatibility Libraries (compat-libraries)
Console Internet Tools (console-internet)
Development Tools (development)
Graphical Administration Tools (graphical-admin-tools)
Legacy UNIX Compatibility (legacy-unix)
Scientific Support (scientific)
Security Tools (security-tools)
Smart Card Support (smart-card)
System Administration Tools (system-admin-tools)
System Management (system-management)
Done
[root@localhost ~]#
```

This shows us all the groups that are available.

```
[root@localhost ~]# yum group list ids kde*
```

Available environment groups:

KDE Plasma Workspaces (kde-desktop-environment)

Done

Some groups are hidden by settings in the configured repositories. For example, on a server, make use of the *hidden* command option to list hidden groups too:

```
[root@localhost ~]# yum list hidden ids kde*
```

Loaded plugins: product-id, subscription-manager

Available Groups:

KDE (kde-desktop)

Done

u can install a package group by passing its full group name, without the groupid part, to the *Group install* command.

For ways of installing the KDE desktop Via Groups.

```
[root@localhost /] yum group install "KDE Desktop"
[root@localhost /] yum group install kde-desktop
[root@localhost /] yum group install @"KDE Desktop"
[root@localhost /] yum group install @kde-desktop
```

As well as four ways of removing the desktop group

```
[root@localhost /] yum group remove "KDE Desktop"
[root@localhost /] yum group remove kde-desktop
[root@localhost /] yum group remove @"KDE Desktop"
[root@localhost /] yum group remove @kde-desktop
```

Now moving on, Yum lets us view the transaction history of every single package that has been installed via YUM. lets run these two commands. This will allow us pinpoint who is installing what. That way, if we see something fishy, we know we have been compromised.

```
[root@localhost /] yum history list
[root@localhost /] yum history list all
```

```
[root@localhost /]# yum history list
Loaded plugins: fastestmirror, langpacks
ID      | Login user      | Date and time    | Action(s)      | Altered
-----|-----|-----|-----|-----
  4     | lu <lu>         | 2016-11-15 01:23 | Install        | 9 <
  3     | lu <lu>         | 2016-11-15 00:46 | Install        | 1 >
  2     | lu <lu>         | 2016-11-15 00:18 | I, U           | 89
  1     | System <unset>  | 2016-11-14 19:34 | Install        | 1354
history list
[root@localhost /]# yum history list all
Loaded plugins: fastestmirror, langpacks
ID      | Login user      | Date and time    | Action(s)      | Altered
-----|-----|-----|-----|-----
  4     | lu <lu>         | 2016-11-15 01:23 | Install        | 9 <
  3     | lu <lu>         | 2016-11-15 00:46 | Install        | 1 >
  2     | lu <lu>         | 2016-11-15 00:18 | I, U           | 89
  1     | System <unset>  | 2016-11-14 19:34 | Install        | 1354
history list
[root@localhost /]#
```

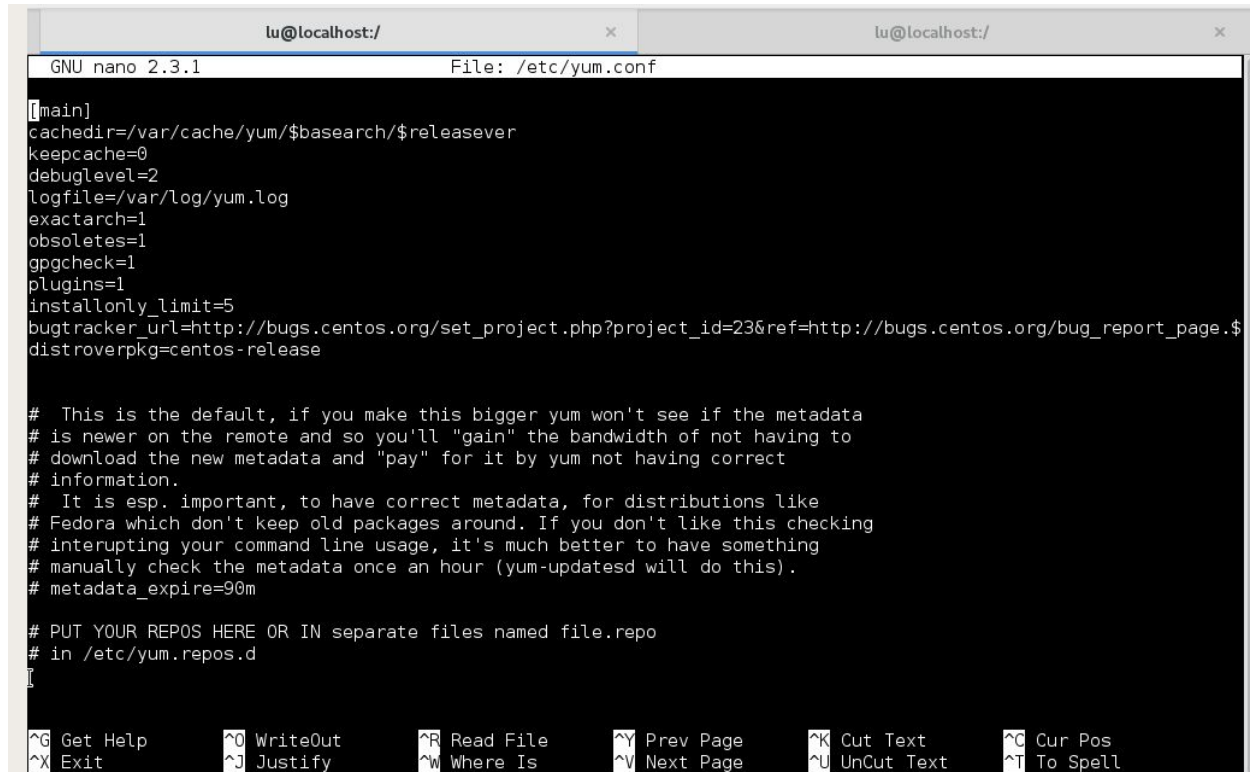
Lastly the main configuration files are located in

```
[root@localhost /] cd yum
```

```
[root@localhost etc]# cd yum
yum/
yum.repos.d/
[root@localhost etc]# cd yum
```


This is the main configuration file for the YUM package manager.

`[root@localhost /] nano /etc/yum.conf`



```
GNU nano 2.3.1 File: /etc/yum.conf

[main]
cachedir=/var/cache/yum/$basearch/$releasever
keepcache=0
debuglevel=2
logfile=/var/log/yum.log
exactarch=1
obsoletes=1
gpgcheck=1
plugins=1
installonly_limit=5
bugtracker_url=http://bugs.centos.org/set_project.php?project_id=23&ref=http://bugs.centos.org/bug_report_page.$
distroverpkg=centos-release

# This is the default, if you make this bigger yum won't see if the metadata
# is newer on the remote and so you'll "gain" the bandwidth of not having to
# download the new metadata and "pay" for it by yum not having correct
# information.
# It is esp. important, to have correct metadata, for distributions like
# Fedora which don't keep old packages around. If you don't like this checking
# interrupting your command line usage, it's much better to have something
# manually check the metadata once an hour (yum-updatesd will do this).
# metadata_expire=90m

# PUT YOUR REPOS HERE OR IN separate files named file.repo
# in /etc/yum.repos.d
I
^G Get Help      ^O WriteOut     ^R Read File    ^Y Prev Page    ^K Cut Text      ^C Cur Pos
^X Exit          ^J Justify      ^W Where Is     ^V Next Page    ^U UnCut Text   ^T To Spell
```

As you can see at the bottom of `/etc/yum.conf` there is a line that says, the repo files are located in `/etc/yum.repos.d`



```
[root@localhost etc]# cd yum.repos.d/
[root@localhost yum.repos.d]# ls
CentOS-Base.repo  CentOS-Debuginfo.repo  CentOS-Media.repo  CentOS-Vault.repo
CentOS-CR.repo   CentOS-fasttrack.repo  CentOS-Sources.repo  google-chrome.repo
[root@localhost yum.repos.d]# nano CentOS-Base.repo
```

So inside the `CentOS-Base.repo` file, there is a list of mirrors that are coming directly from the official CentOS server. As you can see we also have a `google-chrome.repo` file. This was created when we installed the google chrome browser. Its set form automatic updates once you run `yum update`

[root@localhost /] nano /etc/yum.repos.d/CentOS-Base.repo

```
GNU nano 2.3.1 File: CentOS-Base.repo
# CentOS-Base.repo
#
# The mirror system uses the connecting IP address of the client and the
# update status of each mirror to pick mirrors that are updated to and
# geographically close to the client. You should use this for CentOS updates
# unless you are manually picking other mirrors.
#
# If the mirrorlist= does not work for you, as a fall back you can try the
# remarked out baseurl= line instead.
#
#
[base]
name=CentOS-$releasever - Base
mirrorlist=http://mirrorlist.centos.org/?release=$releasever&arch=$basearch&repo=os&infra=$infra
#baseurl=http://mirror.centos.org/centos/$releasever/os/$basearch/
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-7

#released updates
[updates]
name=CentOS-$releasever - Updates
mirrorlist=http://mirrorlist.centos.org/?release=$releasever&arch=$basearch&repo=updates&infra=$infra
#baseurl=http://mirror.centos.org/centos/$releasever/updates/$basearch/
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-7
```

[root@localhost /] nano /etc/yum.repos.d/google-chrome.repo

```
lu@localhost:/etc/yum.repos.d x lu@localhost:/
GNU nano 2.3.1 File: google-chrome.repo

[google-chrome]
name=google-chrome
baseurl=http://dl.google.com/linux/chrome/rpm/stable/x86_64
enabled=1
gpgcheck=1
gpgkey=https://dl.google.com/linux/linux_signing_key.pub
```

As you can see third party repo files are a lot less complicated. YUM is a lot more complicated than APT-GET

That's it for now! Eventually I will cover Pacman and Emerge. I know this chapter is a lot but it's one of the most important fundamental skills.

4.3 Arch Linux Package Management

4.4 Gentoo Package Management