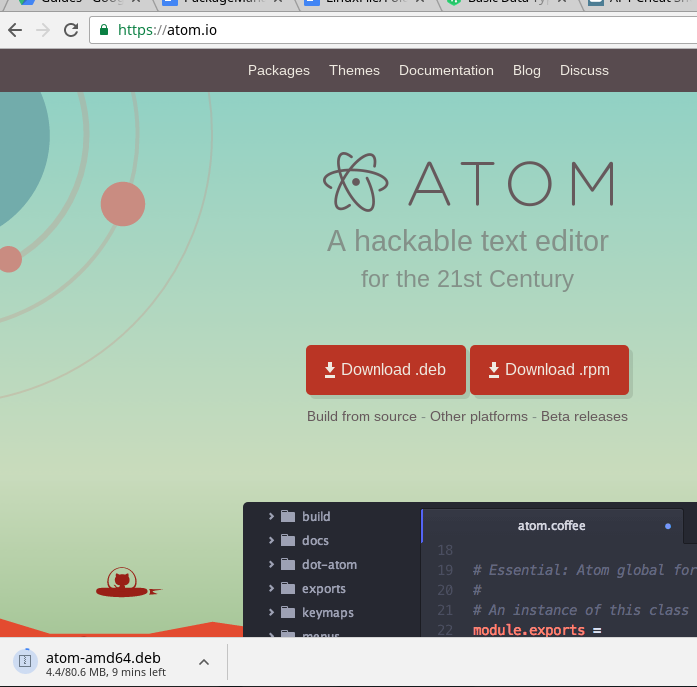
# 4.0 Package Managers

**Package managers** are a way of installing, removing, upgrading software on a Linux system. There are a few very popular package managers that exists, 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.

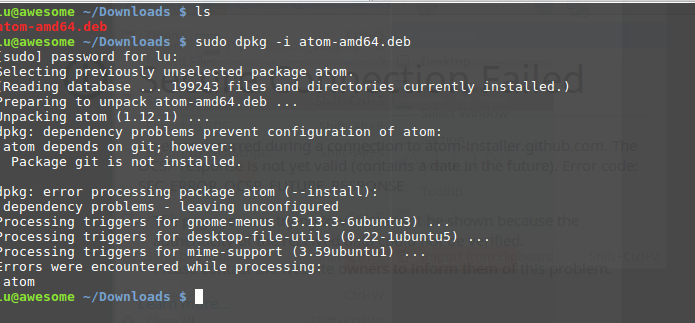
## 4.1 Debian Package Management

**Debian** has a few tools to 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/**](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.



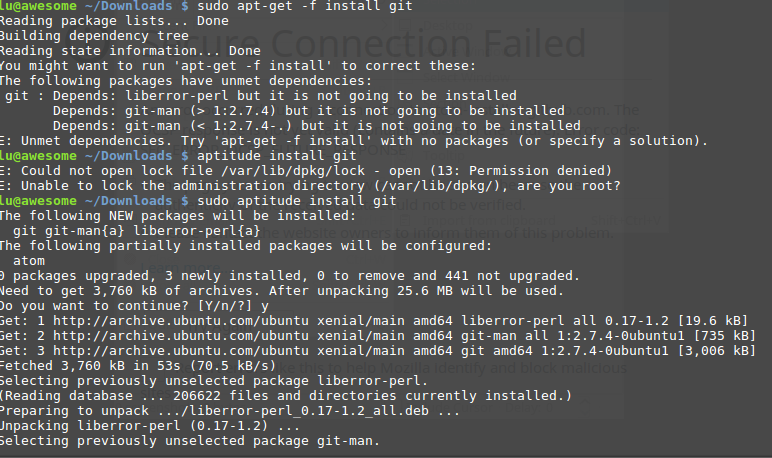
If you get the following error, that is okay, there is a **dependency** problem. A dependency can be a package, library or framework that needs to be installed in order for the software package you need to run and install properly. Let's fix this by running

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!

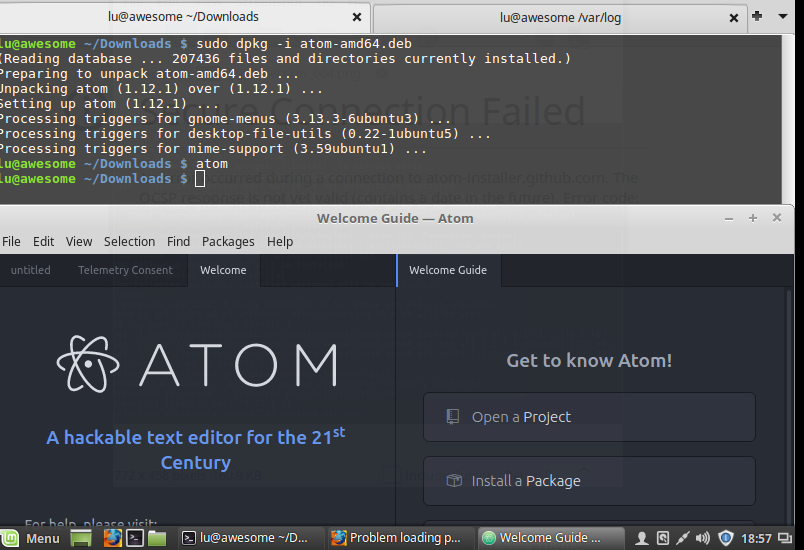
lu@awesome ~/Download $ sudo aptitude install git



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

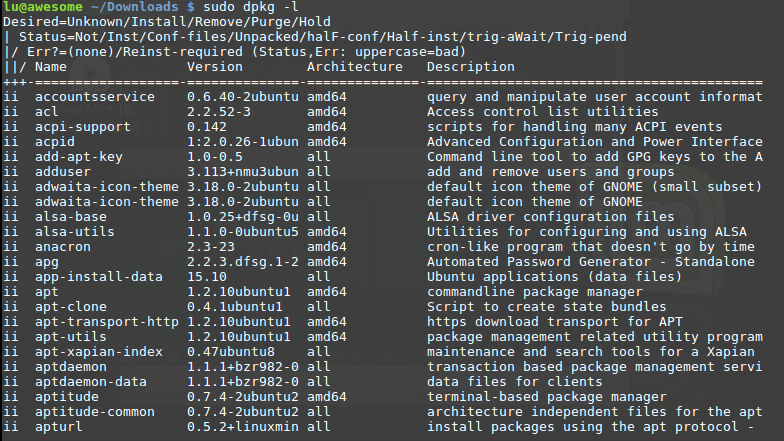


Why would we need to install .deb packages if APT and Aptitude can do this for us? Well there are a lot of third party software that are not available in the official debian repositories , and a .deb file makes it easier than downloading and compiling a package from source. Essentially a .deb package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

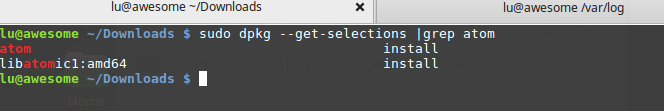


All of the packages will be displayed into a list. Too long of a list right? What if we want to filter these results? Or output them into a file?

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

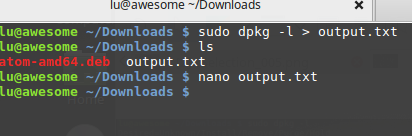
lu@awesome ~/Download $ sudo dpkg --get-selections |grep apache

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



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 ~/Download $ sudo dpkg -l > output.txt



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 -l* 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, lets 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



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:

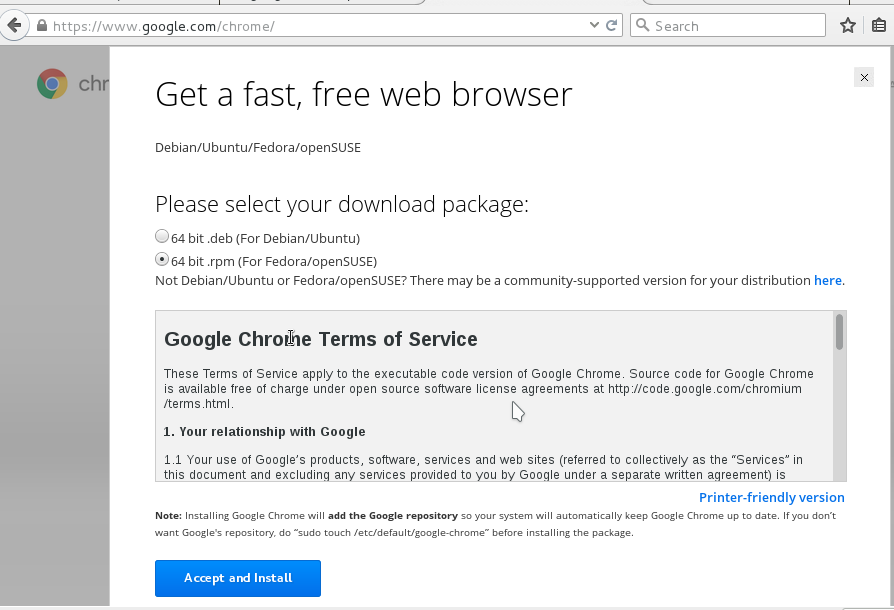
lu@awesome ~/Download $ sudo apt-cache showpkg PACKAGE

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.

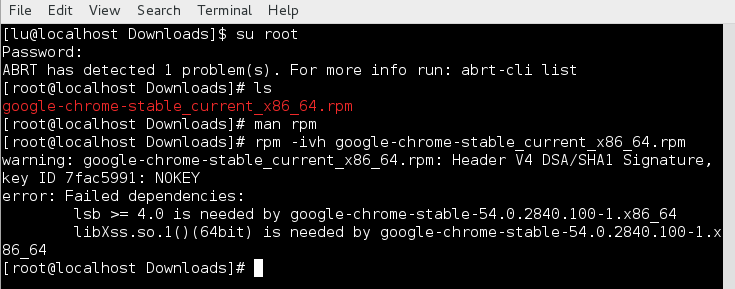
Inside your CentOS virtual machine navigate open up firefox and navigate to the google chrome website and download the .rpm file. Lets install that baby. Since google chrome is not in the official repositories off CentOS.



Navigate to the dir and execute the following command. Switch to root and run

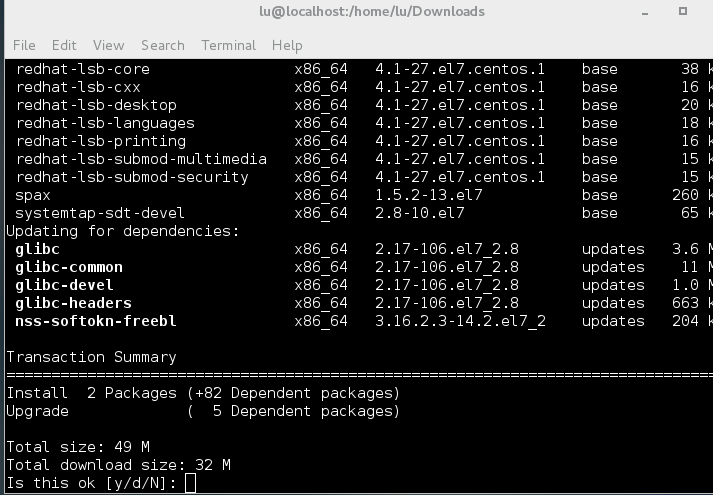
[root@localhost Downloads] rpm -ivh google-chrome-stable\_current\_x86\_64.rpm

*-i* to install , *v* to print verbose (-V to verify) and *h* to print the has.



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

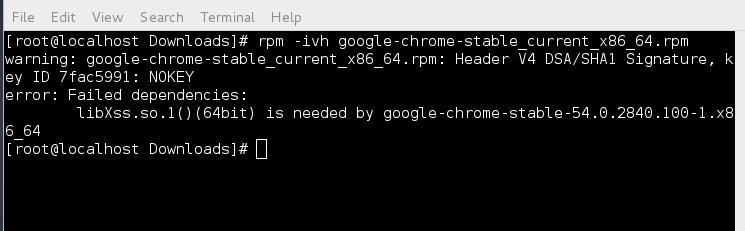
[root@localhost Downloads] yum install lsb libxss.so.1



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



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

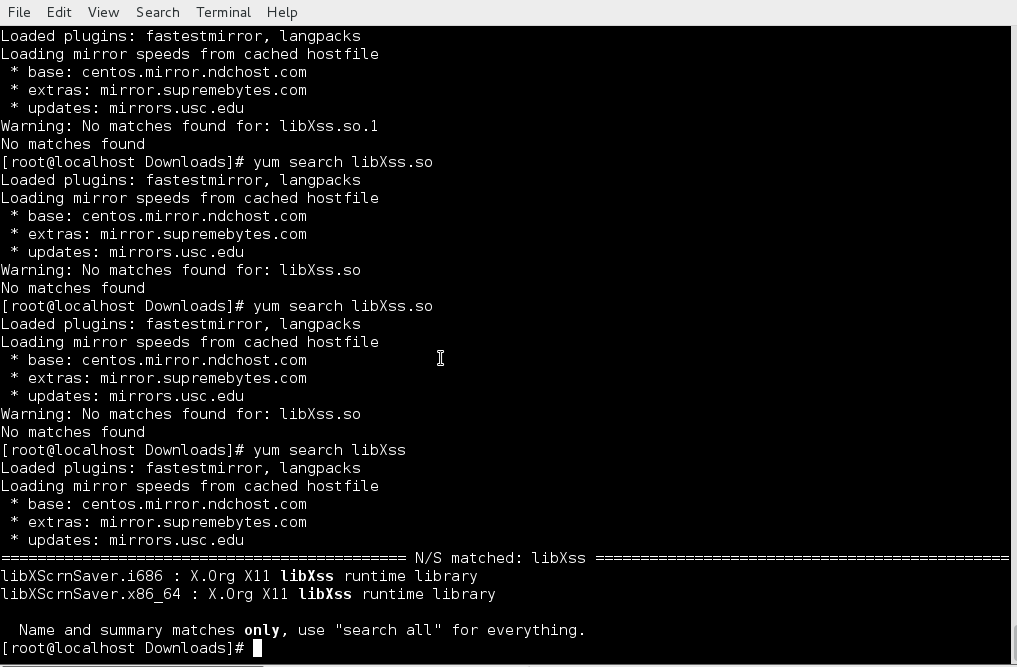
NOTHING will be found. Lets try trimming the search query down

[root@localhost Downloads] yum search libXss.so

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

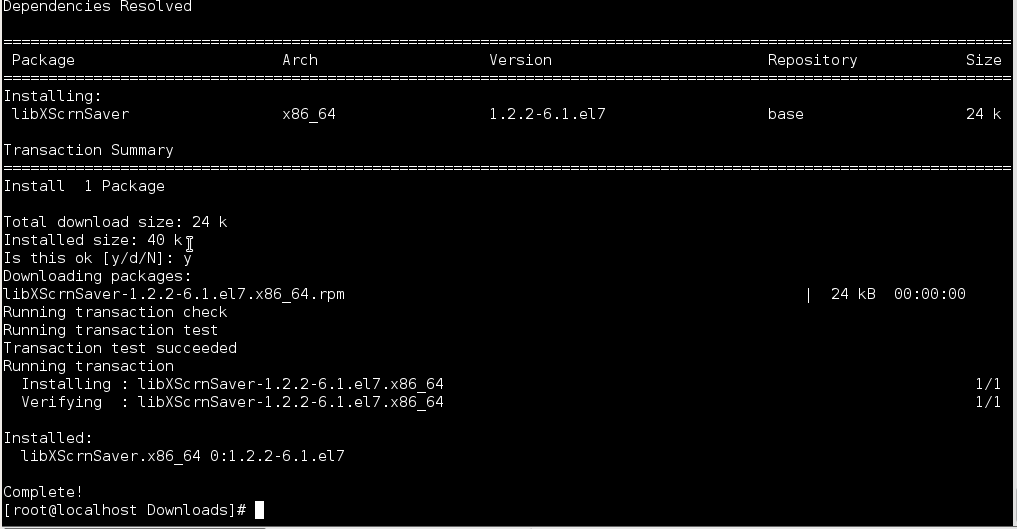
[root@localhost Downloads] yum search libXss

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!



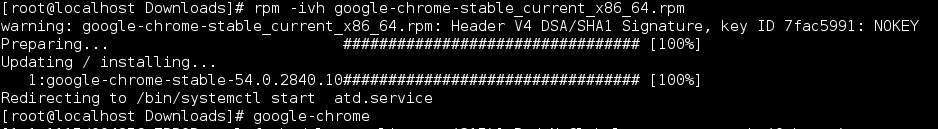
[root@localhost Downloads] yum install libXScrnSaver.x86\_64

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)**.



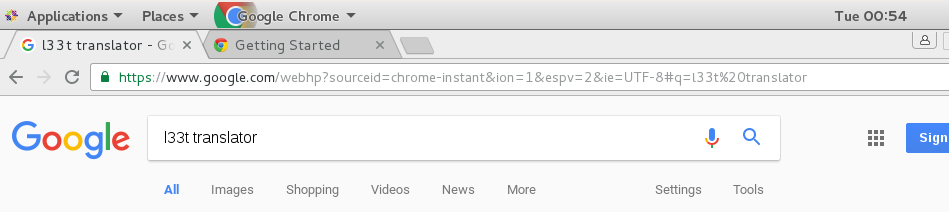
Now Chrome should behave and install properly. Try it one last time

[root@localhost Downloads] rpm -ivh google-chrome-stable\_current\_x86\_64.rpm



**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

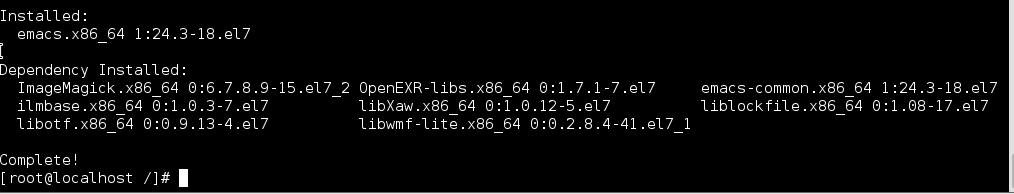
Query a package to find more information about it

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

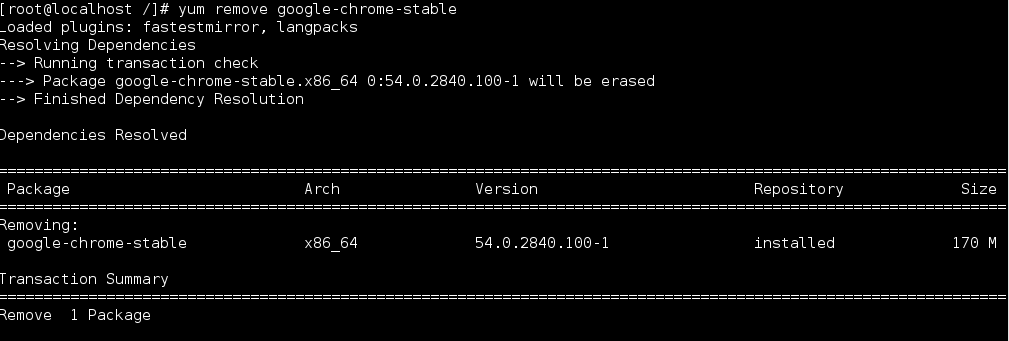


**That's it for RPM!. Let's move on to YUM. we already know how to install a package and we already know how to search for packages. Lets run a couple more examples.**

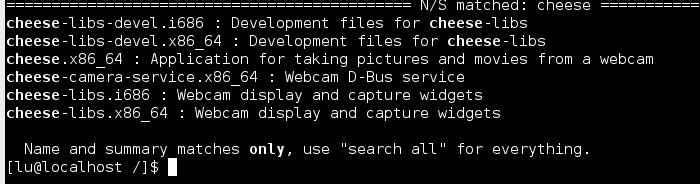
[root@localhost /] yum install emacs



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



[root@localhost /] yum search cheese



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

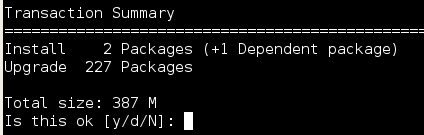
[root@localhost /] yum check-update

This is 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



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 a*rguments.

[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.**

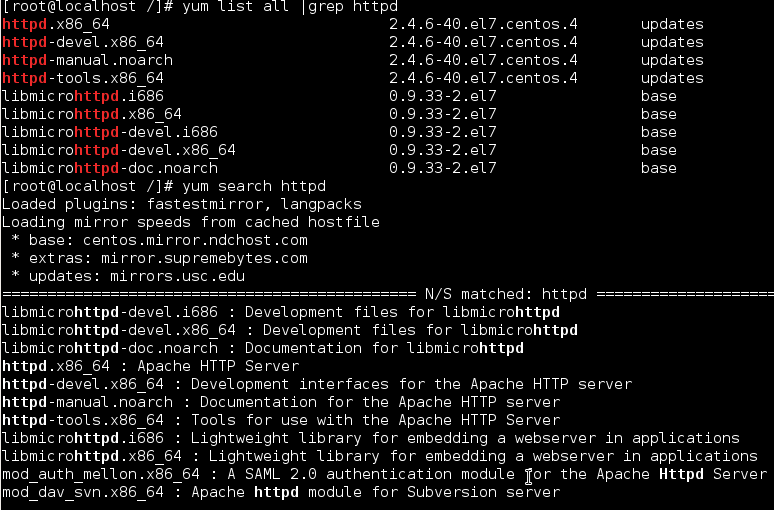
Now lets list all the packages available in the repo’s

[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.

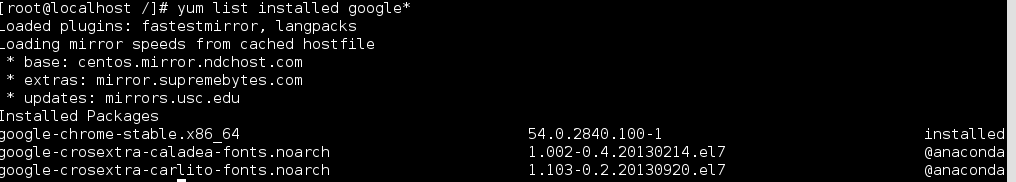
[root@localhost /] yum list all |grep httpd

[root@localhost /] yum search httpd



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\*



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

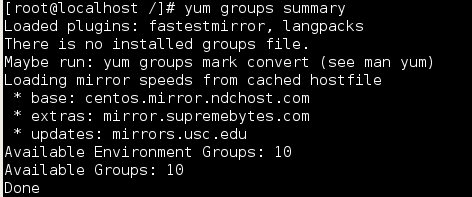
Lets check packages for more info

[root@localhost /] yum info empathy

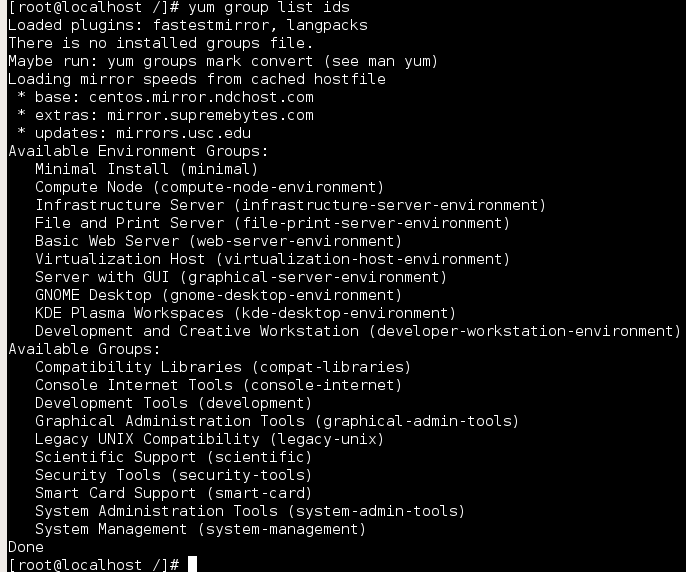


Now the last few things we will be talking about is groups, transaction history, and mirror list. Groups allows us to install packages that are grouped together, such as different desktops and different IDE’s

[root@localhost /] yum groups summary



[root@localhost /] yum list ids



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

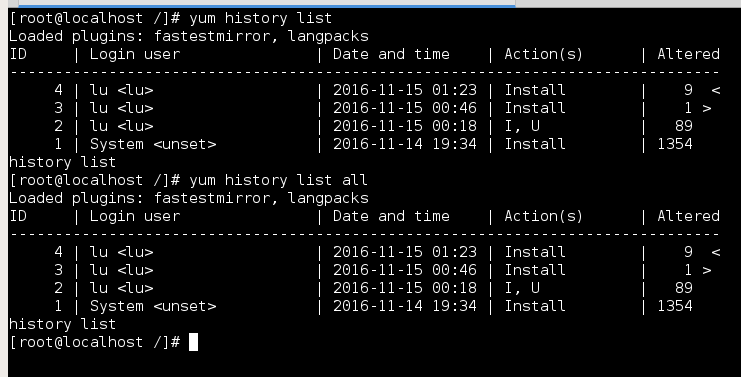
[root@localhost /] yum group remove @”KDE Desktop”

[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.

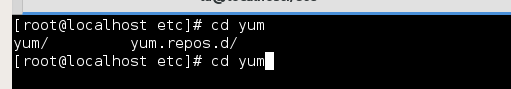
[root@localhost /] yum history list

[root@localhost /] yum history list all



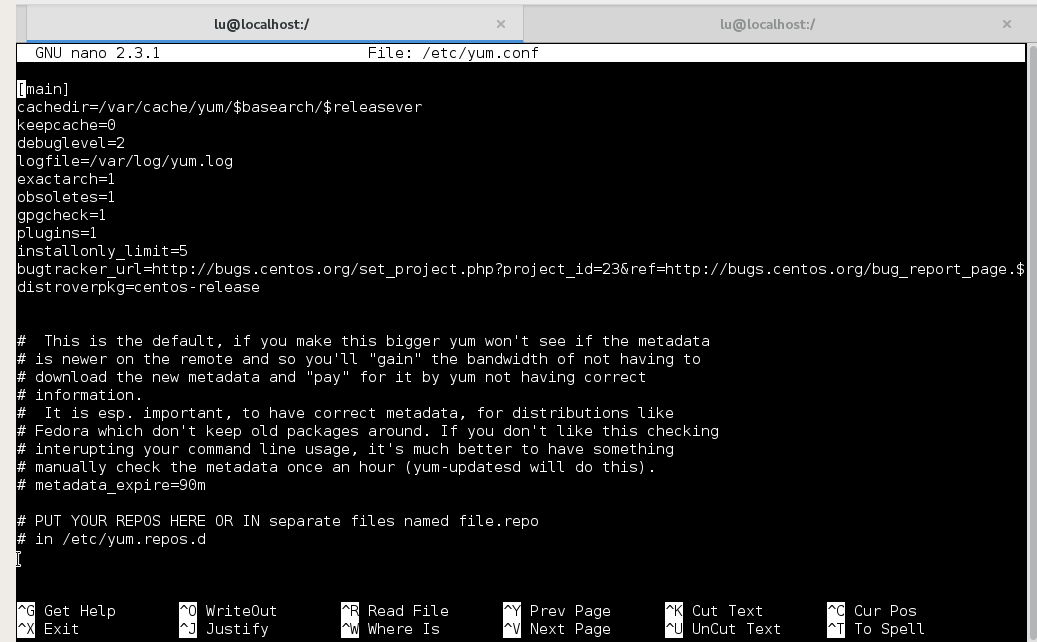
Lasty the main configuration files are located in

[root@localhost /] cd yum

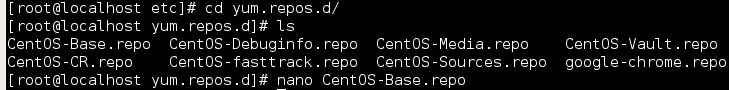


This is the main configuration file for the YUM package manager.

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

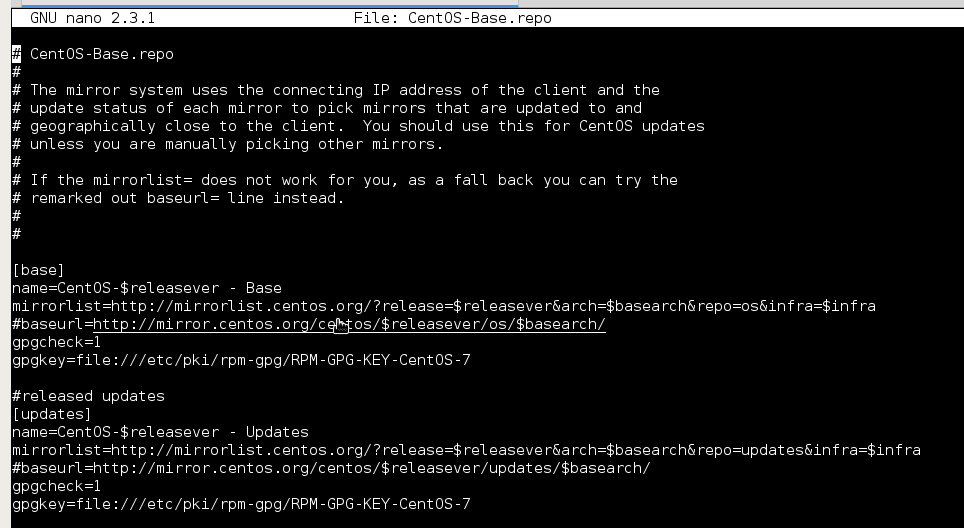


As you can see at the bottom of /etc/yum.conf there is a line that says, the repo files are located in /etc/yum.repos.d

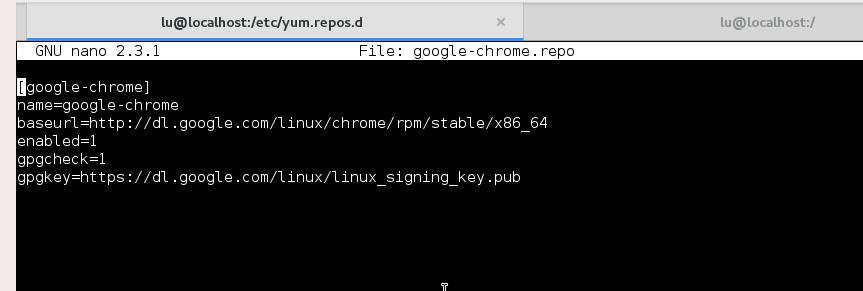


So inside the CentOS-Base.repo file, there is a list of mirrors that are coming directly from the official CentOS server. As you can see we also have a google-chrome.repo file. This was created when we installed the google chrome browser. Its set form automatic updates once you run *yum update*

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



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



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