ProLUG – Patching the system and package management with Dnf, RPM

Required Materials

Rocky 9 Server

Root or sudo command access

EXERCISES (Warmup to quickly run through your system and familiarize yourself)

1. cd ~
2. rpm -qa | more
3. rpm -qa | wc -l
   pick any <name of package> from the above list
4. rpm -qi <name of package>
5. rpm -qa | grep -i imagemagick
6. dnf install imagemagick
   What is the error here? Read it
7. dnf install ImageMagick
   What are some of the dependencies here? Look up the urw-base35 and see what functionality that adds.
8. rpm -qa | grep -i imagemagick
   Why did this work when the other one didn't with dnf?

Math Practice

Some fun with the command line and basic scripting tools. I want you to see some of the capabilities. That are available to you. Your system can do a lot of basic arithmetic for you and this is a very small set of examples.

Check to see if you have bc tool.

rpm –q bc

Install it if you need to

dnf install bc

1. for i in `seq 1 5`; do free | grep -i mem | awk '{print $3}'; done

Collect the 5 numbers (what do these numbers represent? Use free to find out)

2. echo "(79 + 79 + 80 + 80 + 45) / 5" | bc

Your numbers will vary. Is this effective? Is it precise enough?

3. echo "(79 + 79 + 80 + 80 + 45) / 5" | bc -l

Is this precise enough for you?

4. man bc

Read the man to see what the -l option does to bc

It would be astute to point out that I did not have you do bash arithmetic. There is a major limitation of using bash for that purpose in that it only wants to deal with integers (whole numbers) and you will struggle to represent statistical data with precision. There are very useful tools though, and I would highly encourage you to examine them. (http://tldp.org/LDP/abs/html/arithexp.html)

LAB

Log into your Rocky server and become root.

RPM

RPM is the Redhat package manager. It is a powerful tool to see what is installed on your system and to see what dependencies exist with different software packages. This is a toolset that was born of the frustration of “dependency nightmare” where system admins used to compile from source code only to find they had dependencies, which had dependencies, which had dependencies. RPM helps to deconflict and save huge amounts of time and engineering headaches.

1. Using RPM to see installed software information

   Run through these commands and read `man rpm` to see what they do.

   rpm -qi systemd
   Read about the capabilities of systemd

   rpm -q systemd
   query the package given

rpm –qa
query all packages on the system (is better used with | more or | grep)
rpm -qa | grep -i kernel #for example shows all kernels and kernel tools

rpm -qc systemd
List out files, but only show the configuration files

rpm -qi systemd
What good information do you see here? Why might it be good to know that some piece of software was installed last night, if there is now a problem with the system starting last night?

rpm -ql systemd
will list all the files in the package. Why might this be useful to you to know?

rpm -qR systemd
List capabilities on which this package depends

rpm -q -changelog systemd
Probably going to scroll too fast to read. This output is in reverse order.

So let's make it useful with this command
rpm -q -changelog systemd | more
What are some of the oldest entries?
What is the most recent entry?

Is there a newer version of systemd for you to use?
dnf update systemd
If there isn't don't worry about it.

Use `rpm -qa | more` to find 3 other interesting packages and perform `rpm -qi <package>` on them to see information about them.

DNF

Yum comes from a long decrepit version of Linux called Yellow dog. It is originally the Yellowdog Update Manager. It has a very interesting history surrounding the PS3, but that and other nostalgia can be found here: https://en.wikipedia.org/wiki/Yellow_Dog_Linux if you're interested. We're going to use it to update our system. RHEL and CentOS systems look to repositories of software for installation and

updates. We have a base set of them provided with the system, supported by the vendor or open source communities, but we can also create our own from file systems or web pages. We'll be mostly dealing with the defaults and how to enable or disable them, but there are many configurations that can be made to customize software deployment.

1. Checking how dnf is configured and seeing it's available repositories
   cat /etc/dnf/dnf.conf
   has some interesting information about what is or isn't going to be checked. You can include a line here called exclude= to remove packages from installation by name. Where a repo conflicts with this, this takes precedence.

   dnf repolist

   dnf history

2. Checking where repos are stored and what they look like
   ls /etc/yum.repos.d/

   Repos are still stored in /etc/yum.repos.d

   Cat /etc/yum.repos.d/rocky.repo

```
# rocky.repo
#
# The mirrorlist system uses the connecting IP address of the client and the
# update status of each mirror to pick current mirrors that are geographically
# close to the client.  You should use this for Rocky updates unless you are
# manually picking other mirrors.
#
# If the mirrorlist does not work for you, you can try the commented out
# baseurl line instead.

[baseos]
name=Rocky Linux $releasever - BaseOS
mirrorlist=https://mirrors.rockylinux.org/mirrorlist?arch=$basearch&repo=BaseOS-$releasever$rltype
#baseurl=http://dl.rockylinux.org/$contentdir/$releasever/BaseOS/$basearch/os/
gpgcheck=1
enabled=1
countme=1
metadata_expire=6h
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-Rocky-9
…. Output truncated for brevity's sake….
```

   Something you'll find out in the next section looking at repos is that when they are properly defined they are enabled by default. enabled=1 is implied and doesn't need to exist when you create a repo.
3. Let's disable a repo and see if the output changes at all
   [root@rocky1 yum.repos.d]# dnf config-manager --disable baseos

cat /etc/yum.repos.d/rocky.repo

Should now have the line enabled=0 (or false, turned off)

```
[baseos]
name=Rocky Linux $releasever - BaseOS
mirrorlist=https://mirrors.rockylinux.org/mirrorlist?arch=$basearch&repo=BaseOS-$releasever$rltype
#baseurl=http://dl.rockylinux.org/$contentdir/$releasever/BaseOS/$basearch/os/
gpgcheck=1
enabled=0
countme=1
metadata_expire=6h
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-Rocky-9
…. Output truncated for brevity's sake….
```

4. Re-enable the repo and verify the output
   dnf config-manager --enable base

   Cat /etc/yum.repos.d/rocky.repo

   Should now have the line enabled=1 (or true, turned back on)

```
[baseos]
name=Rocky Linux $releasever - BaseOS
mirrorlist=https://mirrors.rockylinux.org/mirrorlist?arch=$basearch&repo=BaseOS-$releasever$rltype
#baseurl=http://dl.rockylinux.org/$contentdir/$releasever/BaseOS/$basearch/os/
gpgcheck=1
enabled=1
countme=1
metadata_expire=6h
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-Rocky-9
…. Output truncated for brevity's sake….
```

5. Installing software you were asked by an application team.
   So someone has asked for some software and assured you it's been tested in similar environments so you go to install it on their system for them.

   See if we already have a version.
   rpm –qa mariadb

   See if dnf knows about it
   dnf search mariadb
   dnf search all mariadb
   What is DNF showing you? What are the differences between these commands based on the output?

   Try to install it
   dnf install mariadb

hit “N”

Make note of any dependencies that are added on top of mariadb (there’s at least one)

What does DNF do with the transaction when you cancel it? Can you compare this to what you might have used before with YUM? How are they different? (You can look it up if you don’t know.)

Ok, install it

dnf -y install mariadb

Will just assume yes to everything you say.

You can also set this option in /etc/dnf/dnf.conf to always assume yes, it’s just safer in an enterprise environment to be explicit.

6. Removing package with dnf

Surprise, the user calls back because that install has made the system unstable. They are asking for you to remove it and make the system back to the recent version.

dnf remove mariadb

hit “N”

dnf –y remove mariadb

this removes mariadb from your system.

But did this remove those dependencies from earlier?

rpm –q <dependency>

rpm –qi <dependency>

How are you going to remove that if it’s still there?

7. Checking where something came from. What package provides something in your system

One of the most useful commands dnf provides is the ability to know “what provides” something. Sar and iostat are powerful tools for monitoring your system. Let’s see how we get them or where they came from, if we already have them. Maybe we need to see about a new version to work with a new tool.

dnf whatprovides iostat

dnf whatprovides sar

Try it on some other tools that you regularly use to see where they come from.

dnf whatprovides systemd

dnf whatprovides ls

dnf whatprovides python

8. Using Dnf to update your system or individual packages

Check for how many packages need update
dnf update
How many packages are going to update?
Is one of them the kernel?
What is the size in MB that is needed?
Hit “N”
Your system would have stored those in /var/cache/dnf
Let’s check to see if we have enough space to hold those
df -h /var/cache/dnf
Is there more free space than there is needed size in MB from earlier? There probably is, but this becomes an issue. You’d be surprised.

Let’s see how that changes if we exclude the kernel
dnf update --exclude=kernel
How many packages are going to update?
Is one of them the kernel?
What is the size in MB that is needed?
Hit “N”

You can update your system if you like. You’d have to reboot for your system to take the new kernel. If you do that you can then redo the grubby portion and the ls /boot/ will show the new installed kernel, unless you excluded it.

9. Using dnf to install group packages

Maybe we don’t even know what we need to get a project going. We know that we need to have a web server running but we don’t have an expert around to tell us everything that may help to make

that stable. We can scour the interwebs (our normal job) but we also have a tool that will give us the base install needed for RHEL or CentOS to run that server.

dnf grouplist

dnf groupinstall “Development Tools”
How many packages are going to update?
  Is one of them the kernel?
  What is the size in MB that is needed?
 Hit “N”
Do you see a pattern forming?

If you install this you’re going to have developer tools installed on the server but they won’t be configured. How would you figure out what tools and versions were just installed? How might you report this for your own documentation and to a security team that keeps your security baselines?