SUMMIT JBoss WORLD

PRESENTED BY RED HAT

LEARN. NETWORK.
EXPERIENCE OPEN SOURCE.

www.theredhatsummit.com

RPM-ifying System Configurations

Paul Waterman
Professional, IT Systems
Motorola, Inc.
June 24, 2010

Introductory Items

Agenda

- Introductory Items
  Housekeeping items like this agenda, who I am, expectations, etc.
- The Scenario
  How system configurations are typically handled and why RPM's are better.
- RPMs: A Few Quick Reminders
  Pointers to RPM resources, and some reminders about how RPMs work.
- Controlling System Services
  How to make sure that services are automatically configured as you want them.
- System Configuration Files
  How to avoid the pitfalls of using RPM's to install system configuration files.

Who is Paul Waterman?

- Motorolan
  I've worked at Motorola, Inc. for 14 years, and am currently the Linux Technical Lead for Midrange Unix Operations.
- Computer Geek
  My first computer was a TRS-80 Model I with 16k of RAM in 1980.
- Unix Guy
  I was first introduced to Unix (ULTRIX) on a DEC MicroVAX II in 1987.
- Linux Fan
  I started using Linux in 1997 to run Magic: The Gathering tournaments.
- Red Hat Admin
  I began using Red Hat personally with Red Hat 5.2, at Motorola with Red Hat 6.2, was certified RHCE in 2004, and certified RHCA in 2009.

Fun Fact:
Ever played Nethack?
I was the principal author of the WCST Nethack Spoilers.

Expectations

- To get the most out of this presentation, you should ...
  - … have system administration skills.
    (RHCE or equivalent skills will best help you grasp the material.)
  - … be familiar with creating RPM's (spec files, etc.).
    (This is not intended to be a general training session on creating RPM's.)

- If you have questions, don't wait – ask them!
  If the question will take too long to answer or take us too far off topic, we'll move on and you can talk to me afterward.

Potential Disappointment

In the interest of being up-front, one topic mentioned in the session description simply couldn't be covered in the time allocation given:

- Constructing files on the fly in your RPM and how to properly handle validation of those files.

The Scenario

Scenario 1

Does this look familiar?

```
#-----------------------------------------------------------------
# Kickstart configuration file post install script

%post

/sbin/chkconfig --level 0123456 service1 off
/sbin/chkconfig --level 35 service2 on

cat > /etc/resolv.conf <<EOF
Domain          mydomain.com
nameserver      192.168.1.1
nameserver      192.168.2.2
EOF
```

Controlling Configuration in Kickstart

Service configurations and system configuration files are often dropped into the %post section of a kickstart configuration file.

- Good: Ensures that the system is configured in a standard way when a system is first loaded.
- Bad: Doesn't address what happens after the system is first loaded.
  - What if somebody installs a service?
  - What if somebody changes a configuration file?
  - What if you need to update the system?

Scenario 2

Does this look familiar?

```
#!/bin/sh
# Run this script any time to configure the system

/sbin/chkconfig --level 0123456 service1 off
/sbin/chkconfig --level 35 service2 on

cat > /etc/resolv.conf <<EOF
Domain          mydomain.com
nameserver      192.168.1.1
nameserver      192.168.2.2
EOF
```

Controlling Configuration via Script

To avoid the one-time nature of kickstarts, a script may be created to manage service configurations and system configuration files. This can even be run by a kickstart.

- Good: Lets you set the system to a known configuration any time you want.
- Bad: Usually isn't automatic.
- Bad: Usually doesn't let you do validation.
- Bad: Usually doesn't address deliberate local changes.

The Solution?

Create an RPM:

- Can be installed during a kickstart.
- Can be installed later.
- Can be updated if and when needed.
- Can automatically respond to system changes.
- Can be validated.
- Can address deliberate local configuration changes.

RPMs: A Few Quick Reminders

Creating an RPM
Some Useful Resources

If you're not familiar with how RPM spec files and rpmbuild work, or would like to like to become more familiar, here are a couple of useful resources:

- Maximum RPM (old and slightly out-of-date, but good): http://www.redhat.com/docs/books/max-rpm/
- Fedora Project RPM Guide: http://tinyurl.com/fedora-rpm-guide

Creating an RPM

Quick Reminder: The Preamble

An RPM spec file always starts with a preamble or introduction section, something like this:

```
#---------------------------------------------------------------------------
# This spec file is Copyright 2010, My Company, Inc.
#---------------------------------------------------------------------------

Summary: My Company general configuration RPM
Name: mycompany-config-gen
Version: 1
Release: 1
License: Copyright 2010, My Company, Inc.
Group: MyCompany/Configs
Packager: Packager Name <my-email@mycompany.com>
BuildArch: noarch

%description
This RPM provides general services and security configuration for My Company.
```

Configuration RPMs are typically not architecture dependent, so set the build arch to "noarch."

Using RPMs

A Quick Reminder About Order – Installs

When installing an RPM, things happen in this order:

1. Execute any applicable %triggerprein scriptlets.
   (%triggerprein scriptlets are ignored prior to RHEL 5.)
2. Execute %pre scriptlet.
3. Install RPM files.
4. Execute %post scriptlet.
5. Execute any applicable %triggerin scriptlets.

Using RPMs

A Quick Reminder About Order – Uninstalls

When installing an RPM, things happen in this order:

1. Execute any applicable %triggerun scriptlets.
2. Execute %preun scriptlet.
3. Remove RPM files.
4. Execute %postun scriptlet.
5. Execute any applicable %triggerpostun scriptlets.

Using RPMs
A Quick Reminder About Order – Upgrades

When upgrading an RPM (including “upgrades” to older versions), things happen in this order:

1. Install the new RPM
2. Uninstall the old RPM

This can be counter-intuitive and can cause problems if you forget this order when creating your RPM scriptlets.

Controlling System Services

Managing Services in an RPM

Take 1: A simple %pre and %post scriptlet

Here's a common first try at managing services via RPM:

```
%post
/sbin/chkconfig --level 01246 sendmail off
/sbin/chkconfig --level 35 sendmail on

%postun
/sbin/chkconfig --level 0123456 sendmail reset

%files
```

If you explicitly specify each run level, you don't have to worry about changing service defaults.

You always have to have a %files section, even if it's blank. Otherwise, the RPM will not build.

Managing Services in an RPM
Take 1: Why it's broken

Remember what happens when upgrading an RPM?

1. Install the new RPM

   During this step, the new RPM's %post scriptlet will run:

   ```
   /sbin/chkconfig --level 01246 sendmail off
   /sbin/chkconfig --level 35 sendmail on
   ```

2. Uninstall the old RPM

   During this step, the old RPM's %postun scriptlet will run:

   ```
   /sbin/chkconfig --level 0123456 sendmail reset
   ```

The net result is that the service will be back to a default state after any upgrade!

Managing Services in an RPM

Take 2: A slightly more complex %post scriptlet

Let's fix that problem in the %postun scriptlet:

```
%post
/sbin/chkconfig --level 01246 sendmail off
/sbin/chkconfig --level 35 sendmail on

%postun
if [ $1 -eq 0 ] ; then
  /sbin/chkconfig --level 0123456 sendmail reset
fi

%files
```

$1 stores the number of versions of this RPM that will be installed after this uninstall completes.

Managing Services in an RPM

Take 2: Why it's still broken

What about the following?

- What happens if the service you're controlling isn't installed? Does your scriptlet fail?
- What happens if the service you're controlling gets installed later? Does the service just use default run states?
- What happens if the service you're controlling gets upgraded? Does that reset the service run states?

Managing Services in an RPM

The solution: Triggers

Triggers are special scriptlets: Their execution depends on another package being installed or uninstalled.

- %triggerprein
  Executes before the containing RPM is installed if the target RPM is already installed. Also executes before the target RPM is installed so long as the RPM containing the trigger remains installed on the system.

  IMPORTANT: This trigger type is only available on newer versions of RPM (e.g., in RHEL 5+). If you create an RPM containing this type of trigger on RHEL 5, the trigger will be gracefully ignored on older systems. If you attempt to create an RPM containing this type of trigger on older RHEL versions, the %triggerprein directive will be taken as a literal string at that point in the spec file.

- %triggerin
  Executes after the containing RPM is installed if the target RPM is already installed. Also executes after the target RPM is installed so long as the RPM containing the trigger remains installed on the system.

Managing Services in an RPM
The solution: Triggers

Additional trigger types:

- %triggerun
  Executes before the RPM containing the trigger is uninstalled if the target RPM is installed on the system.
  Also executes before the target RPM is uninstalled so long as the RPM containing the trigger remains installed on the system.
- %triggerpostun
  Executes after the target RPM is uninstalled so long as the RPM containing the trigger remains installed on the system.
  Does *not* execute when the RPM containing the trigger is uninstalled.

Managing Services in an RPM

Take 3: Using triggers

Triggers provide the most robust solution:

```
%triggerin -- sendmail
/sbin/chkconfig --level 01246 sendmail off
/sbin/chkconfig --level 35 sendmail on

%triggerun -- sendmail
if [ $1 -eq 0 -a $2 -gt 0 ] ; then
  /sbin/chkconfig --level 0123456 sendmail reset
fi

%files
```

$1 stores the number of versions of this RPM that will be installed after this uninstall completes.
$2 stores the number of versions of the target RPM that will be installed after this uninstall completes.

Managing Services in an RPM

Hints: Remember that targets may differ

If you want your configuration RPM to work across multiple different OS distributions and/or versions, keep in mind that you may have to deal with multiple targets. Some quick examples:

- In RHEL 3, the audit service is provided by the laus package. In RHEL 4 and 5 the auditd service is provided by the audit package.
- In RHEL 3, the xfs service is provided by the Xfree86-xfs package. In RHEL 4 and 5 the same service is provided by xorg-x11-xfs.
- In RHEL 3 and 4, xinetd provides the echo and echo-udp services. In RHEL 5, those same services are called echo-stream and echo-dgram, respectively.

Managing Services in an RPM

Hints: You can have complex targets

If you want sendmail to be enabled on RHEL 5, but disabled on anything else, for example:

```
%triggerin -- sendmail
/sbin/chkconfig --level 0123456 sendmail off

%triggerin -- sendmail, redhat-release = 5Client
/sbin/chkconfig --level 35 sendmail on

%triggerin -- sendmail, redhat-release = 5Server
/sbin/chkconfig --level 35 sendmail on

%triggerun -- sendmail
if [ $1 -eq 0 -a $2 -gt 0 ] ; then
  /sbin/chkconfig --level 0123456 sendmail reset
fi
```

Tip:
Triggers are processed in the order in which they appear in the spec file, and multiple triggers can apply. In this example, the first trigger runs on every system that has sendmail installed. The next two triggers run only on RHEL 5 Client and Server systems, respectively, if sendmail is also installed. These triggers run after the first trigger.

Managing Services in an RPM

Advanced: What about multi-level configurations?

How do you handle a situation where you want to be able to have multiple RPMs affecting the same service in a hierarchical manner? For example:

- A general RPM that is installed on all systems and sets defaults for the entire company.
- A site specific RPM that can override the company-wide defaults if necessary.
- An application or group-specific RPM that can override both site settings and company-wide defaults.

Managing Services in an RPM

Advanced: What about multi-level configurations?

You may think that you can just use RPM's built-in dependency management for multi-level configurations, but you can't (at least not easily).

One solution I have found is to maintain an /etc/service-config file:

- Service config RPMs write a line into the file when they're installed.
- Service config RPMs erase that line when they're uninstalled.
- Each line contains a “level” number that determines precedence.
- Each config RPM should contain triggers to reset the service and then run each entry in the service config file in order of precedence whenever the config or service RPM is installed or uninstalled.

Managing Services in an RPM

Advanced: Example multi-level configuration

```
%define level 10
```

This defines a macro: Whenever we use "%{level}" in the rest of the spec file, rpmbuild will substitute "10". We're going to use this to define the precedence level for this RPM. We might then use "20" for a site RPM and "30" for an application RPM.

```
%post
echo "%{level}:sendmail:%{name}-%{version}-%{release}:/sbin/chkconfig --level
0123456 sendmail off; /sbin/chkconfig --level 4 sendmail on" \
  >> /etc/service-config
```

No line wrap here.

Our %post scriptlet will drop a line into /etc/service-config that looks something like this:

```
10:sendmail:svc-gen-3-1:/sbin/chkconfig --level 0123456 sendmail off; /sbin/chkconfig --level 4 sendmail on
```

Managing Services in an RPM

Advanced: Example multi-level configuration

```
%postun
sed -i '/^%{level}:sendmail:%{name}-%{version}-%{release}:/d' /etc/service-config
if [ -x /etc/rc.d/init.d/sendmail ] ; then
  /sbin/chkconfig --level 0123456 sendmail reset
  sort -n /etc/service-config | awk -F: '{if ($2 == "sendmail") { print $NF}}' \
    | sh
fi
```

Our %postun script removes the /etc/service-config line that was inserted by this RPM regardless of whether the uninstall of this RPM is a straight uninstall or part of an upgrade. (We can do that because each line in the /etc/service-config file is specific to the version of the RPM.)

Then, if sendmail is on the system, it sorts the /etc/service-config numerically, pulls out lines where the second field (split on colons) is “sendmail,” and executes the last field on each of those lines.

Managing Services in an RPM

Advanced: Example multi-level configuration

```
%triggerin -- sendmail
/sbin/chkconfig --level 0123456 sendmail reset
sort -n /etc/service-config | awk -F: '{if ($2 == "sendmail") {print $NF}}' | sh
```

This trigger will run if sendmail is installed when this RPM is installed or if sendmail is later installed.

First, it resets the sendmail service settings to their defaults.

Then it will reset the service configuration in the same manner as the %postun script.

Managing Services in an RPM

Advanced: Example multi-level configuration

If you have three levels of RPMs as previously described, you might end up with an /etc/service-config file that looks something like this:

```
20:sendmail:svc-site-3-1:/sbin/chkconfig --level 0123456 sendmail off; /sbin/chkconfig --level 4 sendmail on
30:sendmail:svc-app-3-1:/sbin/chkconfig --level 0123456 sendmail off; /sbin/chkconfig --level 5 sendmail on
10:sendmail:svc-gen-3-1:/sbin/chkconfig --level 0123456 sendmail off; /sbin/chkconfig --level 3 sendmail on
```

Note that the order doesn't matter, because the trigger script sorts the file numerically before processing it. Thus, when executed, the sendmail triggers will run this:

```
/sbin/chkconfig --level 0123456 sendmail off; /sbin/chkconfig --level 3 sendmail on
/sbin/chkconfig --level 0123456 sendmail off; /sbin/chkconfig --level 4 sendmail on
/sbin/chkconfig --level 0123456 sendmail off; /sbin/chkconfig --level 5 sendmail on
```

Managing Services in an RPM

Advanced: Verifying service configuration

If you want your RPM to be able to report any configuration mismatches when “rpm -V” is run, you need to add a verify script to your spec file. For example:

```
%verifyscript
if [ -e /etc/rc.d/init.d/sendmail ] ; then
  BAD=""
  set $(/sbin/chkconfig --list sendmail)
  if [ $2 != "0:off" ] ; then BAD="0" ; fi
  if [ $3 != "1:off" ] ; then BAD=$BAD"1" ; fi
  if [ $4 != "2:off" ] ; then BAD=$BAD"2" ; fi
  if [ $5 != "3:on" ] ; then BAD=$BAD"3" ; fi
  if [ $6 != "4:off" ] ; then BAD=$BAD"4" ; fi
  if [ $7 != "5:on" ] ; then BAD=$BAD"5" ; fi
  if [ $8 != "6:off" ] ; then BAD=$BAD"6" ; fi
  if [ "$BAD" != "" ] ; then
    echo "sendmail run status mismatch for runlevel(s): $BAD" >&2
  fi
fi
```

Managing Services in an RPM

A Quick Note

Have you noticed anything missing in all of the previous examples?

They all use chkconfig to configure in what runlevels the service is turned on and in what runlevels the service is turned off. However, they don't make any attempt to reconcile the current state of the service to the current run level (i.e., they don't start or stop the service).

Doing so is relatively easy for single level service management, but tricky for multi-level configurations, and is left as an exercise for you.

System Configuration Files

Managing Configuration Files in an RPM

Take 1

This should be easy, right? Let's try it for /etc/ntp.conf:

```
Summary: My Company general configuration RPM
Name: config
Version: 1
Release: 1
License: Copyright 2010, My Company, Inc.
Group: MyCompany/Configs
Packager: Packager Name <my-email@mycompany.com>
BuildArch: noarch
Source: %{name}-%{version}.tgz
BuildRoot: %{_builddir}/%{name}-%{version}

%description
This RPM provides config files for My Company.

%prep
%setup

%files
%config(noreplace) %attr(644,root,root) /etc/ntp.conf
```

We need to specify a source file. This .tgz file will include the files we're going to drop onto the system.

Since we're just going to drop files as-is onto the system and don't need to actually build anything for this RPM, we'll use a “cheater” build root. This sets the build root to the directory where %setup will unpack our files. We'll just use those as if they're intalled.

This will preserve local edits during future upgrades.

Managing Configuration Files in an RPM

Take 1

Now we create a .tgz file containing the files going into our RPM. Its contents look like this:

```
% tar tzf /usr/src/redhat/SOURCES/config-1.tgz
config-1/
config-1/etc/
config-1/etc/ntp.conf
```

Managing Configuration Files in an RPM

Take 1

Building the RPM goes smoothly:

```
% rpmbuild -bb /usr/src/redhat/SPECS/config-1.spec
Executing(%prep): /bin/sh -e /var/tmp/rpm-tmp.92476
+ umask 022
+ cd /usr/src/redhat/BUILD
+ LANG=C
+ export LANG
+ unset DISPLAY
+ cd /usr/src/redhat/BUILD

...

+ exit 0
Processing files: config-1-1
Provides: config(config) = 1-1
Requires(rpmlib): rpmlib(CompressedFileNames) <= 3.0.4-1 rpmlib(PayloadFilesHavePrefix) <=
4.0-1
Requires: config(config) = 1-1
Checking for unpackaged file(s): /usr/lib/rpm/check-files /usr/src/redhat/BUILD/config-1
Wrote: /usr/src/redhat/RPMS/noarch/config-1-1.noarch.rpm
```

Managing Configuration Files in an RPM

Take 1: Why it's broken

What happens when we try to install this RPM?

```
# rpm -i config-1-1.noarch.rpm
        file /etc/ntp.conf from install of config-1-1.noarch conflicts with file from package
ntp-4.2.2p1-9.el5_3.2.x86_64
```

Whoops! It turns out that the ntp RPM installed a default or sample ntp.conf file on the system already, so RPM thinks that the ntp RPM “owns” /etc/ntp.conf.

Managing Configuration Files in an RPM

Take 1: Why it's broken

You can install the RPM using the --replacefiles option, but then you end up with *two* RPMs owning the same file:

```
# rpm -qf /etc/ntp.conf
ntp-4.2.2p1-9.el5_3.2
config-1-1
```

You'll also have to remember to use the --replacefiles option when you upgrade either of the two RPMs in the future, or the upgrade will fail. Upgrades can also play havoc with local changes to the config files depending on how each RPM is configured.

Managing Configuration Files in an RPM

The Solution

The solution is to put your file in another location. Then use a trigger to move the original configuration file aside and instead create a symbolic link to your file.

- The original RPM still owns the canonical location for the configuration file.
- Your RPM owns its own configuration file.

Managing Configuration Files in an RPM

Take 2: Using our own location and symlinks

Here's an example of how you might do this:

(Remember that %triggerin gets executed after your RPM is installed if the target is already installed, and after the target gets installed while your RPM is installed.)

```
%triggerin -- ntp
if [ ! -h /etc/ntp.conf -o ! "`readlink /etc/ntp.conf`" = "/mycompany/etc/ntp.conf" ] ; then
  if [ -e /etc/ntp.conf ] ; then
    mv -f /etc/ntp.conf /etc/ntp.conf.orig
  fi
  ln -s /mycompany/etc/ntp.conf /etc/ntp.conf
fi
```

If the config file isn't a symlink or if it is a symlink but the symlink doesn't point to the location of our config file...

If the config file exists, we rename it.

Then we create a symbolic link where the config file should be, and point that symbolic link at our own config file.

Managing Configuration Files in an RPM

Take 2: Using our own location and symlinks

(Remember that %triggerun gets executed before your RPM is uninstalled if the target is installed, and before the target gets uninstalled while your RPM is installed.
The %triggerpostun trigger gets executed after the target is uninstalled while your RPM is installed, but does not run if your RPM is uninstalled.)

```
%triggerun -- ntp
if [ $1 -eq 0 -a $2 -gt 0 -a -e /etc/ntp.conf.orig ] ; then
  mv -f /etc/ntp.conf.orig /etc/ntp.conf
fi

%triggerpostun -- ntp
if [ $2 -eq 0 ] ; then
  rm -f /etc/ntp.conf.rpmsave /etc/ntp.conf.orig
fi
if [ -e /etc/ntp.conf.rpmnew ] ; then
  mv /etc/ntp.conf.rpmnew /etc/ntp.conf.orig
fi
```

If (a) this is the last uninstall of our config RPM, (b) a copy of the ntp RPM will still be installed, and (c) /etc/ntp.conf.orig exists, we'll move that back to /etc/ntp.conf.

If this is the last uninstall of the target, we erase any .rpmsave or .orig config file left on the system.

If there's an .rpmnew config file on the system (caused by an upgrade of the target), we rename it to .orig after the that upgrade completes.

Managing Configuration Files in an RPM

Take 2: Using our own location and symlinks

(Remember that %postun gets executed after our RPM is uninstalled.)

```
%postun
if [ -e /etc/ntp.conf.orig -a -h /etc/ntp.conf -a ! -e "`readlink /etc/ntp.conf`" ] ; then
  mv -f /etc/ntp.conf.orig /etc/ntp.conf
fi
```

If (a) an .orig file exists, (b) the ntp.conf file is a symlink, and (c) the symlink points to a file that exists, we'll move the .orig file back over the top of the sym link.

```
%files
%config(noreplace) %attr(644,root,root) /mycompany/etc/ntp.conf
```

The %file list includes our config file elsewhere on the system.

Managing Configuration Files in an RPM
How It Works

Here's how this solution works:

- If you install your RPM but the target isn't installed:
  Your config file sits there waiting.
- If you later install the target, or if the target is already installed when your RPM gets installed:
  The target's default config file gets moved aside and your config file gets used.
- If you upgrade the target:
  Your config file still gets used.
- If you upgrade your RPM:
  Any local changes you might have made get preserved.
- If you uninstall your RPM:
  The original configuration file file (or the new file if the target has been upgraded in the interim) gets put back in place.

Managing Configuration Files in an RPM

Hints: Complex Targets

Complex targets can be even more useful with config files, because they allow you to create a single RPM that can handle configs for different versions of your target.

For instance:

```
# Trigger for GDM 2.4 config files
%triggerin -- gdm >= 2.4, gdm < 2.5
# Link /etc/X11/gdm/gdm.conf -> /mycompany/etc/X11/gdm/gdm.conf-2.4

# Trigger for GDM 2.6 config files
%triggerin -- gdm >= 2.6, gdm < 2.7
# Link /etc/X11/gdm/gdm.conf -> /mycompany/etc/X11/gdm/gdm.conf-2.6

%files
%config(noreplace) %attr(644,root,root) /mycompany/etc/X11/gdm/gdm.conf-2.4
%config(noreplace) %attr(644,root,root) /mycompany/etc/X11/gdm/gdm.conf-2.6
```

Conclusion

We've covered a lot of material in this presentation, but you should hopefully now know how to better use RPMs to manage your system configurations!

FOLLOW US ON TWITTER

www.twitter.com/redhatsummit

TWEET ABOUT IT

#summitjbw

READ THE BLOG

http://summitblog.redhat.com/