Custom RPMs

For system configuration

Presented by
Tim Klemz
Unix Admin (RHCE), Lifetouch Inc.

Preface

1. I still have lots to learn about RPM creation!
2. RPM's, not unlike scripting, get better with each iteration
3. Lots of resources available – Fedora spec files for examples, RH Summit past presentations..
4. RPMs are a good stepping stone to puppet configuration management

Today's Topics

1. Really quick RPM primer
2. Why should I use RPM's?
3. Example use cases
4. Breakdown of one of our sample system configuration RPMs
5. Things we've Learned along the way..

- RPM Quick Primer

RPM Quick Primer

- Create rpmbuild user / group (do not build as root)
- install rpm package on build server

```
# yum -y install rpm
```

- Create rpm directory structure

```
# mkdir -p /opt/rpmbuild/rpm/
{BUILD,BUILDROOT,RPMS,SOURCES,SPECS,SRPMS}
```

RPM Quick Primer

- Create structure/files for RPM you are making

```
# mkdir -p ~/rpm/SOURCES/helloworld-1.0/var
# echo "hello world" > ~/rpm/SOURCES/helloworld-
1.0/var/helloworld.txt
```

- Create the tar.gz of the SOURCE

```
# cd ~/rpm/SOURCES
# tar cvzf helloworld-1.0.tar.gz
helloworld-1.0/
```

RPM Quick Primer

- Create simple spec file to deploy /var/helloworld.txt

```
# cd ~/rpm/SPECS
# vim helloworld.spec
```

```
Name: helloworld
Version: 1.0
Release: 1
Summary: Places the helloworld.txt file into /var/
License: Proprietary
BuildArch: noarch
Source0: %{name}-%{version}.tar.gz
%install
rm -rf $RPM_BUILD_ROOT                    <---------exists
mkdir -p $RPM_BUILD_ROOT                  <------------------Add
cp -R * $RPM_BUILD_ROOT                   <--------------------Add
%files
/var                          <----path that the file will end up in once installed
```

RPM Quick Primer

- Build rpm

```
# rpmbuild -bb helloworld.spec
```

Once completed, you now have an RPM named helloworld-1.0.noarch.rpm in ~/rpm/RPMS/noarch/, that deploys the /var/helloworld.txt when installed

- Why Should I Use RPM's?

Why RPMs?

- Couldn't configurations be done by post install Satellite snippets?
- What if you are already using configuration management tool (puppet, chef, RH Satellite)?

Why RPMs?

- Kickstart(Satellite) Snippets are only done on build
- What if there is a change you wish to make to your snippet? How do you apply that the already deployed systems?
- What happens is somebody changes a configuration file?

Why RPMs?

- It's true, configuration management tools can accomplish these same things, but...
  - Although many benefits exist when using a mature configuration management tool... who is at this point?
  - Handle deliberate local configuration changes?
  - RPM's could bridge the gap, and help with logic to later convert tasks to puppet, chef etc...

Why RPMs?

- Can be installed during a kickstart
- Can be installed anytime afterwords
- Can be updated and applied when needed
- Can setup to address deliberate local configuration changes

RPM Use Cases

RPM Use Cases

- Biggest use case, Oracle database server builds!
- Standardized application server builds
- Reduce build time & hand over servers quicker to end users
  - Handle all configuration changes and server specific application installs in a single “prep” rpm

RPM Use Cases

- Handle Configuration of system builds
  - User and access configuration
  - Additional applications required (Many requirements for Oracle DB systems!)
  - Application specific kernel parameters
  - Ulimit settings
- When user access changes, RPM can be updated and applied with a simple “yum -y update”

Example RPM

Example RPM

- These examples are used to prepare almost all system builds
- Files to be deployed exist in tarball for RPM in the SOURCES directory
- We are utilizing RHDS(ldap) for our user management and access control, but proxy/role accounts are local

What We've Learned

What We've Learned

- Be careful with the “%un” sections in spec file!
- RPM's are not shell scripts, don't treat them that way
- Make sure your logic is sound
- Only use “Requires” for packages that your RPM truly require to operate

What We've Learned

- Knowing the order of steps for an RPM package update/upgrade is imperative
  - First installs upgrade RPM, then uninstalls original
- This has major implications!
  - Uninstall directives in spec file are not just used when uninstalling (ie removing) an RPM package
  - %postun says remove oracle user? Upgrade will run and last step will be remove oracle user
- Overall last steps taken are the “%un” sections from the package you are updating!

What We've Learned

- When doing “yum update”, order of steps seems like the wrong order(shown on previous slide). This really impacts doing updates when there are commands in the %un sections
- DO NOT REMOVE A USER IN AN %un SECTION!

What We've Learned

- Initial RPM's we were making were performing all sorts of seds / echos as if it was a shell script
  - No benefit to use RPM if doing this
  - Difficult if not impossible to upgrade
  - Not able to easily return to previous state when removing the package

What We've Learned

- Ensure your logic is sound!
  - If there is a mistake in your RPM (especially uninstall sections) you can't fix it once the RPM is installed.
  - Solid logic ensures RPM installs/upgrades and removes without scary error messages
- Test your RPM for multiple scenarios before rolling it out (install, update, removal)

Summary

- RPMs can be beneficial when used to perform system configurations
- Extremely helpful when not (yet) using a configuration management tool
- When combined with Kickstart, it can makes server setup a breeze

Resources

- http://www.redhat.com/promo/summit/2010/presentations/summit/opensource-for-it-leaders/thurs/pwaterma-2-rpm/RPM-ifying-System-Configurations.pdf
- http://docs.fedoraproject.org/en-US/Fedora_Draft_Documentation/0.1/html/RPM_Guide/index.html

Questions?

Contact:
tklemz@lifetouch.com