RED HAT
SUMMIT

LEARN. NETWORK.
EXPERIENCE OPEN SOURCE.

June 11-14, 2013
Boston, MA

redhat.

FROM CONVENTIONAL RPMS TO SOFTWARE COLLECTIONS

Bohuslav “Slavek” Kabrda

Software Engineer, Red Hat

Thomas Cameron, RHCA, RHCDS, RHCSS, RHCVA, RHCX
Chief Architect, Western US, Red Hat

June 14th, 2013

redhat.

Agenda

Introduction

- Who am I?
  - Why should you care?

What are we going to do today?

- Build an RPM. Badly!
- There is a method to the madness!
  - If we build an RPM that does things RPMs don't normally do, we learn what RPM is **supposed** to do.
  - Don't do this in the real world!

Agenda

Assumptions

- About you
- About why on Earth we're doing something this silly
- Why “the way we've always done it” is wrong.
- About the software we're installing

Agenda

The steps

- Install the necessary packages to build a compile environment
- Create a dedicated build account
- Create your build environment
- Set up a gpg key
- Get the source for the software you're going to build
- Build a shell script to set environment variables for the RPM

Agenda

The steps

- Patch the software
- Create an init script
- Create a spec file
  - Edit the preamble
  - Redefine some global macros
  - Edit the %prep section and test it
  - Edit the %build section and test it
  - Edit the %install section and test it
  - Set up the %files section

Agenda

The steps

- Create a spec file (continued)
  - Set up the %changelog section
- Build a binary RPM
- Build a source RPM
- Build both RPMs
- Sign the RPM
- Import the GPG key and test the signature
- Install the RPM
- Test the application

Agenda

The steps

- Modify some files and verify the package
- Remove the package

Introduction

Who am I?

- Thomas Cameron
  - Regional Chief Architect, Western US
  - Nothing but an old sysadmin
  - Academic background in Criminal Justice, former police officer who changed careers 20 years ago
  - I am NOT a developer

Introduction

Who am I?

- Why should you care?
  - Because I had to figure this out from a non-programmer point of view. If you're new to RPMs, this session is for you!
  - If you're an old hand, stick around for the next half!

What are we going to do today?

Build an RPM.

What are we going to do today?

Build an RPM. Badly!

What are we going to do today?

Build an RPM. Badly!

- We're going to build an RPM to install buggy software in a non-standard location. Don't do this in the real world!

There is a method to the madness!

- Forcing RPM to do non-standard stuff teaches you how it's supposed to work, too...

Assumptions

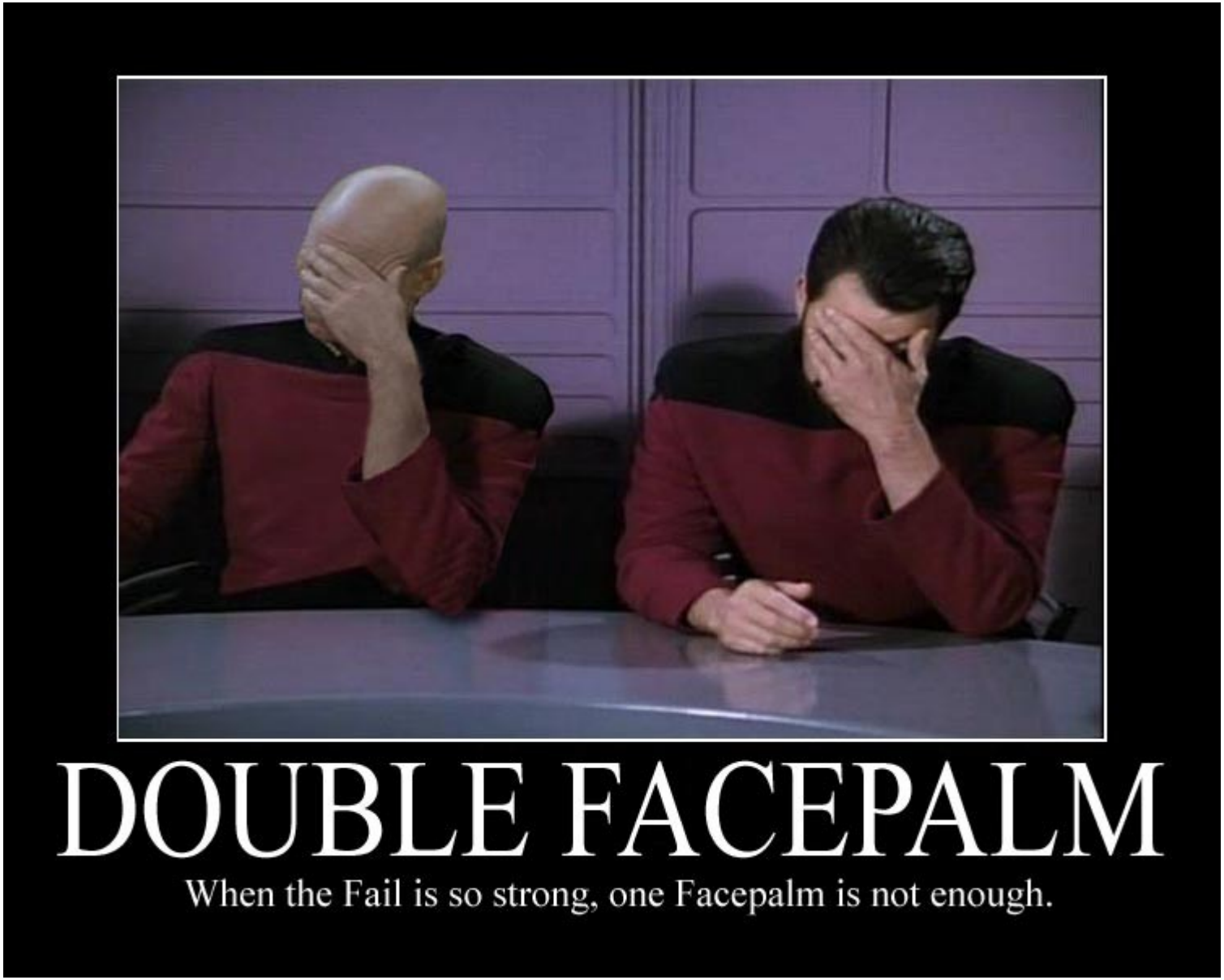
About you:

- You're at least peripherally familiar with compiling software. You do not need to be a professional programmer, but I assume you understand the basics.

Assumptions

About why we're doing this silly thing.

- I get asked all the time about installing third party or internally developed software.
- I hear all the time “Building RPMs is too hard. Oh, and we don't install where Red Hat does, we do it in /opt just like we did with [proprietary, expensive *nix]. I just compile it on my workstation, tar it up, scp it to my machines, unpack the tarball, hack /etc/profile for environment variables, start it with rc.local, and we're good! We've always done it that way!”


DOUBLE FACEPALM
When the Fail is so strong, one Facepalm is not enough.

“The way we've always done it” is wrong.

That is NOT enterprise software management.

- “for i in host1 host2 host3; do scp myfile.tgz root@$i:; done” is NOT an acceptable method of enterprise software deployment!
- How do you verify versions for your deployed software?
- How do you consistency check it?
- How do you know that the next time you build from tarball, the same configure-time and compile-time options are used?
- What if you get hit by a beer truck? Can the next poor slob they hire figure out what you did?

Here's the system! Have fun with that!

Assumptions

About the software we're installing

- Your boss has told you to install a newer version of Apache httpd than comes with RHEL 6.
- Your boss has told you to put in in /opt/apache so it's just like all the other *nix systems.

For the sake of brevity, we're not going through all the iterations you'd normally go through figuring this out. I will tell you that:

- There's a bug that prevents “make install DESTDIR=[dir]” from working.

To the labs

There are 21 labs. It sounds like a lot, but each one only takes a minute or two.

Get the labs.txt file from
http://people.redhat.com/tcameron/Summit2013/labs.txt

Lab 1: Install the necessary packages to build a compile environment

Import the Red Hat GPG public key used to verify RPMs

- rpm --import /etc/pki/rpm-gpg/RPM-GPG-KEY-redhat-release

Install the “Software Development” package group.

- yum groupinstall "Development Tools"

Install the rpmdevtools package

- yum install rpmdevtools

Lab 2: Create a dedicated build account

Never create RPMs as root.

- Software is written by humans, and therefore has bugs.
- You do not want a bug in autoconf or the like to cause your compile to write files all over the filesystem.
- As a non-root user, that can't happen.

Lab 2: Create a dedicated build account

Create a user account called "makerpm," and set it's password to “redhat”

PLEASE do not make any other changes to your system! Other labs depend on these systems working correctly!

- Log into your system as root, with a password of summit2013
- useradd makerpm
- passwd makerpm

Log out

Lab 3: Create your build environment

rpmdevtool-setuptree will create an RPM build environment in your home directory

- ~/.rpmmacros for any personal customizations you want rpmbuild to use
- ~/rpmbuild
  - ~/rpmbuild/RPMS where finished packages go
  - ~/rpmbuild/SRPMS for source RPMs
  - ~/rpmbuild/SPECS for spec files
  - ~/rpmbuild/SOURCES for sources
  - ~/rpmbuild/BUILD a scratch/build directory

Lab 3: Create your build environment

Log in as makerpm.

- Check your home directory with
  - ls -al
- Run the command:
  - rpmdev-setuptree

Lab 3: Create your build environment

Log in as makerpm.

- Check again
  - ls -al
- Note the new file and directory
- Comment out the %__arch_install_post line from the .rpmmacros file.

Lab 4: Set up a gpg key

You should always digitally sign your packages. In fact, yum will not, by default, allow the installation of unsigned RPMs.

You should consider a two-stage process for building packages, where a trusted signor is different from the builders.

Execute the following command to create your keys:

- gpg --gen-key

Lab 4: Set up a gpg key

Export your public key (or “armor”), to be imported into the RPM databases on systems upon which you'll be installing RPMs.

Execute the following command to export this public key

- gpg --export -a > MY-PUBLIC-KEY

Lab 4: Set up a gpg key

Add your gpg key ID to your .rpmmacros file.

Your public gpg key ID can be determined by running the command:

- gpg --list-keys

```
[makerpm@host236 ~]$ gpg --list-keys
/home/makerpm/.gnupg/pubring.gpg
--------------------------------
pub   2048R/ACCB6E72 2013-06-10
uid                  Make RPM <makerpm@example.com>
sub   2048R/A03A2A81 2013-06-10
```

Lab 4: Set up a gpg key

Your .rpmmacros will look something like this:

```
%_topdir      %(echo $HOME)/rpmbuild
%_smp_mflags  -j3
#%__arch_install_post /usr/lib/rpm/check-rpaths /usr/lib/rpm/check-buildroot
%_signature gpg
%_gpg_name ACCB6E72
```

Lab 5: Get the source for the software you're going to build

Download it from http://archive.apache.org/dist/httpd/httpd-2.2.17.tar.gz

- wget http://archive.apache.org/dist/httpd/httpd-2.2.17.tar.gz

Lab 6: Build a shell script to set environment variables for the RPM

Because we're putting httpd in a non-standard location, we need to set environment variables for PATH and MANPATH.

Setting these in e.g. /etc/profile is a bad idea – it might get overwritten on an update.

RHEL provides /etc/profile.d for scripts to be run at login. Create a shell script which the RPM will install there.

Lab 6: Build a shell script to set environment variables for the RPM

The completed apache.sh script will look like this:

```
#!/bin/bash
PATH="/opt/apache/bin:$PATH"
export PATH
MANPATH="/opt/apache/man:$MANPATH"
export MANPATH
```

Save it in ~/rpmbuild/SOURCES

Lab 7: Patch the software

We never, ever modify the pristine source when building RPMs.

The pristine source must be included in the build environment.

We can patch it, but we must define those patches so anyone who uses the package can understand exactly how it has been changed.

Lab 7: Patch the software

The version of Apache we're using, 2.2.17, has a bug in the expat Makefile.in. The developer forgot to enable DESTDIR.

It's a simple, three line change in httpd-2.2.17/srclib/apr-util/xml/expat/Makefile.in

In order to create the patch, extract the source, make a copy of the file you're going to patch with a meaningful extension (I chose .orig):

- cp ./httpd-2.2.17/srclib/apr-util/xml/expat/Makefile.in ./httpd-2.2.17/srclib/apr-util/xml/expat/Makefile.in.orig

Lab 7: Patch the software

Once you've made the change, use gendiff to create the patch, and write it to SOURCES:

- gendiff httpd-2.2.17/ .orig > ~/rpmbuild/SOURCES/make_DESTDIR.patch

Lab 7: Patch the software

The patch should look like this:

```
diff -up httpd-2.2.17/srclib/apr-util/xml/expat/Makefile.in.orig httpd-2.2.17/srclib/apr-util/xml/expat/Makefile.in
--- httpd-2.2.17/srclib/apr-util/xml/expat/Makefile.in.orig   2013-06-11 00:55:17.853001685 -0400
+++ httpd-2.2.17/srclib/apr-util/xml/expat/Makefile.in        2013-06-11 00:55:43.520994059 -0400
@@ -85,9 +85,9 @@ check: tests/runtests
 install: installlib

 installlib: $(LIBRARY) $(APIHEADER)
-       $(mkinstalldirs) $(libdir) $(includedir)
-       $(LIBTOOL) --mode=install $(INSTALL) $(LIBRARY) $(libdir)/$(LIBRARY)
-       $(INSTALL_DATA) $(APIHEADER) $(includedir)
+       $(mkinstalldirs) $(DESTDIR)$(libdir) $(DESTDIR)$(includedir)
+       $(LIBTOOL)  --mode=install $(INSTALL) $(LIBRARY) $(DESTDIR)$(libdir)/$(LIBRARY)
+       $(INSTALL_DATA) $(APIHEADER) $(DESTDIR)$(includedir)

 $(LIBRARY): $(LIB_OBJS)
        $(LINK_LIB) $(LIB_OBJS)
```

Lab 8: Create an init script

Since this is a system daemon, we need to control it. rpmdevtools includes a utility, rpmdevtools-newinit, which creates the basic framework of a start/stop script.

- cd ~/rpmbuild/SOURCES
- rpmdev-newinit httpd

You'll need to configure the chkconfig settings (start order, stop order, and runlevels) and description.

Lab 8: Create an init script

The chkconfig section will look something like this:

```
# chkconfig: - 85 15
# description: custom httpd for Acme
```

Note that these lines **are** commented out. chkconfig reads them and determines what symlinks to create under what directories in /etc/rc.d/init.d

Lab 8: Create an init script

Set the exec path to:

- exec="/usr/sbin/httpd"

Lab 8: Create an init script

Set the start stanza to:

```
# if not running, start it up here, usually something like "daemon $exec"
daemon $exec
retval=$?
```

Lab 8: Create an init script

Set the stop stanza to:

```
# stop it here, often "killproc $prog"
killproc $exec
retval=$?
```

Lab 9: Create a spec file

In the ~/rpmbuild/SPECS directory, create a spec file:

- cd ~/rpmbuild/SPECS
- rpmdev-newspec httpd

Lab 10: Edit the preamble

The preamble contains basic information about the RPM. What the package name is, where the source code is from, what the licenses is, and so on.

You can also override default RPM building macros in the preamble.

To see the default macros, run the command:

- rpm --showrc

Lab 10: Edit the preamble

Set the package name to the name the developer gave it. It's “httpd,” don't change it.

- Name: httpd

Lab 10: Edit the preamble

Define the version. This the version the software author gave it.

- Version: 2.2.17

Lab 10: Edit the preamble

Define the release number. This is your build number. You must increment this every time you release your build of the software!

Note that in this case, the number is [n]%{?dist}.

The %{?dist} is an RPM macro which will be expanded to “el6.”

- Release: 1%{?dist}

Lab 10: Edit the preamble

Write a summary. Keep it short but descriptive. The full description will be done later.

- Summary: Apache httpd for Acme Widget

Lab 10: Edit the preamble

Define which package group your RPM will belong to.

The group should be set based on the contents of /usr/share/doc/rpm-4.8.0/GROUPS

- Group: System Environment/Daemons

Lab 10: Edit the preamble

Define which license your package is released under. In this case, it's the Apache Software License v2.0

- License: ASL 2.0

Lab 10: Edit the preamble

Define the URL from which you can learn more about the software.

URL: http://httpd.apache.org

Lab 10: Edit the preamble

Define the first source file. In this case, it is the pristine httpd-2.2.17.tar.gz file we downloaded earlier.

- Source0: http://archive.apache.org/dist/httpd/httpd-2.2.17.tar.gz

Lab 10: Edit the preamble

Define the next source file. In this case, the source file is not going to be compiled, but it is still a source file for the RPM.

- Source1: apache.sh

Lab 10: Edit the preamble

Define the first patch file. In this case, it's the make_DESTDIR.patch file:

- Patch0: make_DESTDIR.patch

Lab 10: Edit the preamble

Define the packages which are required to build the software. This is usually defined in the README or INSTALL file that comes in the source tarball, the project web site, or you can just run through cycles of ./configure until it completes.

- BuildRequires: libtool, autoconf

Lab 10: Edit the preamble

Define the packages which are required to run the software once it's installed. Again, the README or INSTALL file, other included docs, or the web site should have this.

- Requires: perl

Lab 10: Edit the preamble

Define the build root – the location in which the compiled output will be installed. This is the “root” of the RPM. Files which will wind up in /bin on a system when the RPM is installed will be compiled and placed into the build root before they are packaged.

Lab 10: Edit the preamble

For some reason, the spec file created by rpmdev-newspec does not include this information. You can cheat by running "vim bogus.spec," and copy and paste the BuildRoot line from there. bogus.spec should not be saved, we're just using it to get the correct syntax for the BuildRoot line.

- BuildRoot: %(mktemp -ud %{_tmppath}/%{name}-%{version}-%{release}-XXXXXX)

Lab 10: Edit the preamble

Add a description to the preamble. This is free-form text. Keep it short.

- %description
- This is a custom build of Apache httpd for Acme Widget

Lab 10: Edit the preamble

So now, the preamble should look like this:

```
Name:          httpd
Version:       2.2.17
Release:       1%{?dist}
Summary:       Apache httpd for Acme Widget

Group:         System Environment/Daemons
License:       ASL 2.0
URL:           http://httpd.apache.org
Source0:       httpd-2.2.17.tar.gz
Source1:       apache.sh
Patch0:        make_DESTDIR.patch

BuildRequires:  make, libtool
Requires:      perl
BuildRoot:     %(mktemp -ud %{_tmppath}/%{name}-%{version}-%{release}-XXXXXX)

%description
This is a custom build of Apache httpd for Acme Widget
```

Lab 11: Redefine some macros

As mentioned earlier, there are some global settings defined in the system macros file, /usr/lib/rpm/macros.

You can see these macros by running:

- rpm --showrc

You can override these macros in the spec file. You can define them for the scope of the running rpmbuild process, or make them global. It is generally better to make them global.

Lab 11: Redefine some global macros

Per Panu Matilainen, RPM hacker and packager extraordinaire, the rules are basically:

- Always create parametrized macros with %define.
- Inside %{ } blocks you need to use %global to define global macros.
- %define'd macro is evaluated at time of use, %global macro is evaluated at time of definition (otherwise it might have references to macros that have gone out of scope at time of use)

Lab 11: Redefine some global macros

Because we're doing the equivalent of:

- ./configure --prefix=/opt/apache

... we need to see what the equivalent macro for --prefix is. Run the command:

- rpm --showrc | grep prefix

You will see that the default for the _prefix macros is /usr

- _prefix /usr

Lab 11: Redefine some global macros

In order to set the _prefix macro globally, we'll add this to the preamble of the spec file:

- %global _prefix /opt/apache

Lab 11: Redefine some global macros

Through trial and error, you'll find that even with _prefix defined, the system configuration directory gets set to /etc. A grep of rpm --showrc shows us that the macro we need to define globally is _sysconfdir. If we want **everything** to go under /opt, we need to define the macro globally like this:

- %global _sysconfdir /opt/apache/etc

Lab 11: Redefine some global macros

Through trial and error, you'll find that even with _prefix defined, the man pages get placed under /usr/share/man. If you want them under /opt/apache, set the macro globally like this:

- %global _mandir /opt/apache/man

Lab 11: Redefine some global macros

Through trial and error, you'll find that even with _prefix defined, the documentation gets placed under /usr/share/doc. To change that, set the following macro globally:

- %global _defaultdocdir /opt/apache/doc

Lab 11: Redefine some global macros

Now the preamble should look like this:

```
Name:          httpd
Version:       2.2.17
Release:       1%{?dist}
Summary:       Apache httpd for Acme Widget

Group:         System Environment/Daemons
License:       ASL 2.0
URL:           http://httpd.apache.org
Source0:       http://archive.apache.org/dist/httpd/httpd-2.2.17.tar.gz
Source1:       apache.sh
Source2:       httpd.init
Patch0:        make_DESTDIR.patch

BuildRequires:  libtool, autoconf
Requires:      perl
BuildRoot:     %(mktemp -ud %{_tmppath}/%{name}-%{version}-%{release}-XXXXXX)

%global _prefix /opt/apache
%global _sysconfdir /opt/apache/etc
%global _mandir /opt/apache/man
%global _defaultdocdir /opt/apache/doc

%description
This is a custom build of Apache httpd for Acme Widget
```

Lab 12: Edit the %prep section and test it

The prep section unpacks any compressed files and patches them.

We'll add a patch entry to the %prep section. We'll call out patch0 from the preamble:

- %prep
- %setup -q
- %patch0 -p1

Lab 12: Edit the %prep section and test it

You can test that the %prep section works by running the command:

- rpmbuild -bp [specfile]

Lab 13: Edit the %build section and test it

Since we added the globals above, there is nothing we need to do to this section. The %configure macro will expand to:

- ./configure --build=x86_64-redhat-linux-gnu --host=x86_64-redhat-linux-gnu --target=x86_64-redhat-linux-gnu --program-prefix= --prefix=/opt/apache --exec-prefix=/opt/apache --bindir=/opt/apache/bin --sbindir=/opt/apache/sbin --sysconfdir=/opt/apache/etc --datadir=/opt/apache/share --includedir=/opt/apache/include --libdir=/opt/apache/lib64 --libexecdir=/opt/apache/libexec --localstatedir=/var --sharedstatedir=/var/lib --mandir=/opt/apache/man --infodir=/usr/share/info

Lab 13: Edit the %build section and test it

Test using:

- rpmbuild -bc [spec]

Lab 14: Edit the %install section and test it

Fix the %install section. Since we're inserting the apache.sh script into the RPM outside of the compile process, we need to make the /etc/profile.d directory in the RPM build root and install the script there. Since we defined the apache.sh file as Source1 in the preamble, we'll just refer to it as %{SOURCE1} below.

Lab 14: Edit the %install section and test it

Since we're inserting a start/stop script into the RPM outside of the compile process, we need to make the /etc/rc.d/init.d directory in the RPM build root, and install the script there. Since we defined the httpd.init file as “Source2” in the preamble, we just use the macro %{SOURCE2}, below

Lab 14: Edit the %install section and test it

Also, we need to create a directory in the RPM build root for logs, so we create the directory $RPM_BUILD_ROOT/var/logs. Alternatively, we could have patched the httpd.conf file so the logs are created in /var/log.

```
%install
rm -rf $RPM_BUILD_ROOT
make install DESTDIR=$RPM_BUILD_ROOT
mkdir -p $RPM_BUILD_ROOT/etc/profile.d
install -m 755 %{SOURCE1} $RPM_BUILD_ROOT/etc/profile.d
mkdir -p $RPM_BUILD_ROOT/etc/rc.d/init.d
install -m 755 %{SOURCE2} $RPM_BUILD_ROOT/etc/rc.d/init.d
mkdir -p $RPM_BUILD_ROOT/var/logs
```

Lab 14: Edit the %install section and test it

Note that rpmbuild *will* error at this point, since it will attempt to install all the files which are built to the RPM build root. Execute the command:

- rpmbuild -bi httpd.spec 2> error.log

Lab 14: Edit the %install section and test it

If you look at the error.log file, you'll see all the unpackaged files. They all begin with three white spaces, followed by /opt/apache. There are over 1200 of them! Unfortunately, there are several duplicates. To get an accurate list of the files you need to add to the %files section of your spec file, you can run the command:

- grep "   \/opt" error.log | awk '{ print $1 }' | sort | uniq > files

Lab 15: Edit the %files section and test it

All of the README and LICENSE type files from the root of the extracted source file can be added to the %doc section:

- %doc ABOUT_APACHE CHANGES INSTALL LAYOUT LICENSE NOTICE README README.platforms README-win32.txt ROADMAP VERSIONING

Lab 15: Edit the %files section and test it

Now we need to add the files which were unpackaged from Lab 14 and add them to your spec file. DON'T DO THIS NEXT STEP, this is just for discussion!

You could do something like execute the command:

- cat files >> httpd.spec

... then edit the httpd.spec file and move all those files appended to the end to the %files section. Your spec file will be over 1200 lines, though.

Lab 15: Edit the %files section and test it

```
%config /opt/apache/etc/extra/httpd-autoindex.conf
%config /opt/apache/etc/extra/httpd-dav.conf
...
... lots of files ...
...
%config /opt/apache/etc/original/extra/httpd-vhosts.conf
%config /opt/apache/etc/original/httpd.conf
```

Lab 15: Edit the %files section and test it

Additionally, you can tell RPM which files are documentation by marking them as %doc:

```
%doc /opt/apache/etc/extra/httpd-manual.conf
%doc /opt/apache/etc/original/extra/httpd-manual.conf
...
... lots of files ...
...
%doc /opt/apache/share/manual/vhosts/name-based.html.ko.euc-kr
%doc /opt/apache/share/manual/vhosts/name-based.html.tr.utf8
```

But who wants to have a spec file that is over 1200 lines long?

Lab 15: Edit the %files section and test it

Fortunately, RPM allows you to glob names in the spec file. So instead of listing every single file, you can reduce your %files section to:

```
%files
%defattr(-,root,root,-)
%doc ABOUT_APACHE CHANGES INSTALL LAYOUT LICENSE NOTICE
README README.platforms README-win32.txt ROADMAP VERSIONING
%config /opt/apache/etc/extra/
%config /opt/apache/etc/original/
%config /opt/apache/etc/httpd.conf
%config /opt/apache/etc/magic
%config /opt/apache/etc/mime.types
%doc /opt/apache/man/
/opt/apache/bin
/opt/apache/include/
/opt/apache/libexec/
/opt/apache/sbin/
/opt/apache/share/
/opt/apache/lib64/
/etc/
/var/
```

Lab 15: Edit the %files section and test it

Now test this with:

- rpmbuild -bi httpd.spec

It should exit with 0

Lab 16: Set up the %changelog section

The %changelog section will also be updated every time you redistribute the RPM.

To get the correct date for the %changelog entry, execute the command:

- date "+%a %b %e %Y"

The entry will look something like this:

```
%changelog
* Sat Jun  8 2013 "Make RPM <makerpm@example.com>"
- First build
```

Lab 16: Set up the %changelog section

Test that the RPM builds.

Execute the command:

- rpmbuild -bb httpd.spec

```
...
Checking for unpackaged file(s): /usr/lib/rpm/check-files
/home/makerpm/rpmbuild/BUILDROOT/httpd-2.2.17-1.el6.x86_64
Wrote: /home/makerpm/rpmbuild/RPMS/x86_64/httpd-2.2.17-1.el6.x86_64.rpm
Wrote: /home/makerpm/rpmbuild/RPMS/x86_64/httpd-debuginfo-2.2.17-1.el6.x86_64.rpm
Executing(%clean): /bin/sh -e /var/tmp/rpm-tmp.3xMbpZ
+ umask 022
+ cd /home/makerpm/rpmbuild/BUILD
+ cd httpd-2.2.17
+ rm -rf /home/makerpm/rpmbuild/BUILDROOT/httpd-2.2.17-1.el6.x86_64
+ exit 0
```

Lab 17: Build the source RPM

Execute the command:

- rpmbuild -bs httpd.spec

```
[makerpm@host236 SPECS]$ rpmbuild -bs httpd.spec
Wrote: /home/makerpm/rpmbuild/SRPMS/httpd-2.2.17-1.el6.src.rpm
```

Lab 18: Build the binary and source RPMs

Execute the command:

- rpmbuild -ba httpd.spec

You should see something like this:

```
[makerpm@host236 SPECS]$ rpmbuild -ba httpd.spec
Executing(%prep): /bin/sh -e /var/tmp/rpm-tmp.YbkMNU
+ umask 022
+ cd /home/makerpm/rpmbuild/BUILD
...
... lots of compile-y build-y stuff
...
Checking for unpackaged file(s): /usr/lib/rpm/check-files
/home/makerpm/rpmbuild/BUILDROOT/httpd-2.2.17-1.el6.x86_64
Wrote: /home/makerpm/rpmbuild/SRPMS/httpd-2.2.17-1.el6.src.rpm
Wrote: /home/makerpm/rpmbuild/RPMS/x86_64/httpd-2.2.17-1.el6.x86_64.rpm
Wrote: /home/makerpm/rpmbuild/RPMS/x86_64/httpd-debuginfo-2.2.17-1.el6.x86_64.rpm
Executing(%clean): /bin/sh -e /var/tmp/rpm-tmp.4gjUZc
+ umask 022
+ cd /home/makerpm/rpmbuild/BUILD
+ cd httpd-2.2.17
+ rm -rf /home/makerpm/rpmbuild/BUILDROOT/httpd-2.2.17-1.el6.x86_64
+ exit 0
```

Lab 19: Sign the RPM

Execute the command:

- rpm -v --checksig /home/makerpm/rpmbuild/RPMS/x86_64/httpd-2.2.17-1.el6.x86_64.rpm

Lab 19: Sign the RPM

You should see something similar to this:

```
[makerpm@host236 SPECS]$ rpm -v --checksig /home/makerpm/rpmbuild/RPMS/x86_64/httpd-
2.2.17-1.el6.x86_64.rpm
/home/makerpm/rpmbuild/RPMS/x86_64/httpd-2.2.17-1.el6.x86_64.rpm:
    Header SHA1 digest: OK (5790196e5bbddd3f4617d3036b3c69a44cd74674)
    MD5 digest: OK (5b75584d5e81207818f3ac5988d0c4f7)
```

Notice that there is no RSA GPG signature on this RPM.

Lab 19: Sign the RPM

Now sign the RPM. Execute this command:

- rpm --resign /home/makerpm/rpmbuild/RPMS/x86_64/httpd-2.2.17-1.el6.x86_64.rpm

Now when you check the signature, you will see a new signature, but "NOKEY:"

```
/home/makerpm/rpmbuild/RPMS/x86_64/httpd-2.2.17-1.el6.x86_64.rpm:
    Header V4 RSA/SHA1 Signature, key ID 43c45aaa: NOKEY
    Header SHA1 digest: OK (5790196e5bbddd3f4617d3036b3c69a44cd74674)
    V4 RSA/SHA1 Signature, key ID 43c45aaa: NOKEY
    MD5 digest: OK (5b75584d5e81207818f3ac5988d0c4f7)
```

Lab 19: Sign the RPM

As root, import the key:

- rpm --import /home/makerpm/MY-PUBLIC-KEY

Now check the signature of the RPM again:

```
[root@host236 ~]# rpm -v --checksig /home/makerpm/rpmbuild/RPMS/x86_64/httpd-2.2.17-1.el6.x86_64.rpm
/home/makerpm/rpmbuild/RPMS/x86_64/httpd-2.2.17-1.el6.x86_64.rpm:
    Header V4 RSA/SHA1 Signature, key ID 43c45aaa: OK
    Header SHA1 digest: OK (5790196e5bbddd3f4617d3036b3c69a44cd74674)
    V4 RSA/SHA1 Signature, key ID 43c45aaa: OK
    MD5 digest: OK (5b75584d5e81207818f3ac5988d0c4f7)
```

Lab 20: Install the RPM

As root, execute:

- yum install /home/makerpm/rpmbuild/RPMS/x86_64/httpd-2.2.17-1.el6.x86_64.rpm

Lab 20: Install the RPM

Start the service:

- /etc/rc.d/init.d/httpd.init start

You should see a green [OK]. Check to see that the process is running:

```
[root@host236 ~]# ps ax | grep httpd
30350 ?        Ss     0:00 /opt/apache/sbin/httpd
30352 ?        S      0:00 /opt/apache/sbin/httpd
30353 ?        S      0:00 /opt/apache/sbin/httpd
30354 ?        S      0:00 /opt/apache/sbin/httpd
30355 ?        S      0:00 /opt/apache/sbin/httpd
30356 ?        S      0:00 /opt/apache/sbin/httpd
```

Lab 21: Modify the RPM

RPM is good for checking the consistency of installed packages.

We'll modify a configuration file of our httpd package and the test it.

We're going to add in an extra, commented line to our configuration file:

- echo -e "\n# " >> /opt/apache/etc/httpd.conf

Lab 21: Modify the RPM

Now use rpm -V to verify our RPM

- [root@host236 ~]# rpm -V httpd

You'll notice that RPM warns us the the size (S), md5sum (5) and time (T) are all modified on this file:

S.5....T. c /opt/apache/etc/httpd.conf

Lab 22: Remove the RPM

Remove the RPM. You'll see that the changed config file gets kept. Run these commands:

- service httpd.init stop
- yum remove httpd

Lab 22: Remove the RPM

You should see something similar to this:

```
[root@host236 ~]# service httpd.init stop
Stopping httpd:                                            [  OK  ]
[root@host236 ~]# yum remove httpd
Loaded plugins: product-id, rhnplugin, security, subscription-manager
...
... lots of remove-y stuff
...
  Erasing    : httpd-2.2.17-1.el6.x86_64                                  1/1
warning: /opt/apache/etc/httpd.conf saved as /opt/apache/etc/httpd.conf.rpmsave
  Verifying  : httpd-2.2.17-1.el6.x86_64                                  1/1

Removed:
  httpd.x86_64 0:2.2.17-1.el6

Complete!
```

Questions?

Thank you!

Please fill out the evaluation form!

- If you liked today's session, give the forms to the fine folks at the room entrance.

Thank you!

Please fill out the evaluation form!

- If you liked today's session, give the forms to the fine folks at the room entrance.
- If you did not like the session, please bring the forms to me so I can “review” them and learn from your feedback.