RPM Building

What to Expect?

- System Setup to build an RPM
- Steps to build an RPM
- Testing the built RPM

Master Steps

- Setting up the build system
- Collating the package source
- Writing the .spec file
- Actually building the rpm
- Testing the built rpm

System Setup

- Creating a separate login, say rpmbuild
  - To avoid security hazards
- Create a package directory
  - mkdir <pkg_dir>
- Create the following subdirectories
  - BUILD: Place for building by rpmbuild
  - RPMS with i386, i686, noarch: For built binary package
  - SOURCES: For the original software sources
  - SPECS: For .spec file
  - SRPMS: For built source rpm
  - tmp: For temporary files (Optional)
- Install the following commands
  - rpmbuild: Typically part of rpm-build package
  - Other Optionals: rpmlint, gpg

Steps to Build an RPM

- Collating the package sources
  - Typically as a .tgz file
- Writing a spec file
  - Basic structure
  - Building the package
  - Cleaning the package
  - Installing the package
  - Uninstalling the package
- Building using rpmbuild

Locate your Templates

- Create the .tgz of sources
  - View the 'sources' target in the makefile
  - Output: SOURCES/pkg_cmd-1.0.tgz
- Spec file & the macros
  - Example: SPECS/pkg_cmd-1.0.spec
- Building the rpm
  - View the 'build' target in the makefile
  - Output: RPMS/pkg_cmd-1.0-1.i386.rpm

Spec Sections

- Introduction
  - Defines, Keywords, %description
- %prep – Unpack source code
- %build – Compile code
- % install – Install code onto build machine
- %clean – Clean up code
- %files – List of files to use
- %changelog – Change log of the code

%file specific Macros

- %defattr – Default attributes
  - file attributes, owner, group, dir attributes
- %doc – Documentation files
- %exclude – Exclude these files

- Useful Path Macros
  - %{_prefix}
  - %{_bindir}, %{_sbindir}, %{_libdir}
  - %{_datadir}, %{_sysconfdir}
  - %{_mandir}, %{_infodir}
  - %{_gamesbindir}, %{_gamesdatadir}

Additional Spec Sections

- %pre – Execute before install
- %post – Execute after install
- %preun – Execute before uninstall
- %postun – Execute after uninstall
- Option to these: -p <interpreter>

Signing an RPM

- Assuming that a GPG key is there
- Add the following macros in the .spec file
  - %_signature gpg
  - %_gpg_path */path/to/.gnupg*
  - %_gpg_name *name lastname (comment) <email>*
    - 'gpg --list-keys' to get the value
  - %_gpgbin /usr/bin/gpg

  Build with --sign option to rpmbuild
- Alternatively, an RPM can be signed as follows
  - rpm --addsign <rpm_package>

  After putting the above macros in ~/.rpmmacros

Interesting Spec Macros

- Triggers on operation with other packages
  - %triggerin <pkg> – On installing <pkg>
  - %triggerun <pkg> – On uninstalling <pkg>
  - %triggerin <pkg> – After uninstalling <pkg>
- Details about all
  - /usr/lib/rpm/macros
- Place for repeated macros
  - ~/.rpmmacros

Building the RPM

- Using rpmbuild
  - rpmbuild <build_options> [options] <spec_file>
- <build_options>
  - -bs, bb, -ba
  - -bp (%prep), -bc (%build), -bi (%install)
  - -bl (list check from %files)
- A very clean way
  - rpmbuild -ba --rmspec --rmsource
  - rpmbuild --rebuild

Testing the RPM Build(ing)

- Are rpms created in their directories?
- Are the rpms with the correct names?
- Is the rpm info correct? Issued by
  - rpm -qlivp --changelog <rpm_package>
- Linting the rpms or even the spec files
  - rpmlint [-i] <rpm_package | spec_file>
- Install Tests
  - Expected files at expected places & privileges
  - Binaries executable, Documentation accessible
  - Various installs, uninstalls
    - On different machines
    - Without required packages

Backup

Options to setup

- -c – Create upper directory first
  - Useful for archive without a parent directory
- -D – Does not delete the directory
  - Useful in later setups, if multiple
- -T – Override the default behaviour
  - Rather specified by -a 0, -b 0, etc
- -n <name> - <name> what source unpacks to
  - Useful if different from the source name

Generating a GPG key

- Generate a key using gpg --gen-key
- Listings (provides <UID>)
  - gpg --list-sigs
  - gpg --list-keys
- Generating a public key
  - gpg --armor --export "<UID>" > my.key.file.asc
- Publishing the public key
  - gpg --keyserver pgp.mit.edu --send-key "<UID>"

What all have we learnt?

- System Setup to build an RPM
- Steps to build an RPM
  - Collating the sources
  - Writing the spec file
  - Building the rpm
- Testing the built RPM

Any Queries?