Software Collections

Marcela Mašláňová

February 2013

Agenda

- what are Software Collections
- problems to solve
- tools
- packaging

Software Collections

- A structural definition for an application or toolset that is independent of the OS.
- It might be a package or set of dependent packages.
- Collections can provide several parallel-installable versions of software.

Running more versions

- Common practice for more versions of software is to install them into /usr/local or a place defined by company policy usually into /opt.
- Sysadmins use
  - symlinks
  - NFS shares

Running more versions

To solve the multiple-versions problem in one distribution several approaches are used:

- environment modules
- languages tools:
  - perlbrew
  - rvm

Problems to solve

- Developers might be interested in exact version of software – eg. Ruby.
  - some developers prefer one version for longer time
    - especially if they maintain project for more than one distribution
  - other developers prefer the latest & greatest

Problems to solve

- Portability across different versions of distributions.
- Packaging or deployment on system with conflicting software.

Software Collections can offer

- Develop the app on the same set of packages on different releases of distribution.
- Port & test apps with new version.
- Package your own rpms into collection for deployment.
- Goal:
  - Easier packaging & deployment on various distribution releases.

Success stories

- Development Toolset product – support C/C++ development on RHEL-5 and RHEL-6 in Software Collections.
  - is-your-c-development-team-...
- OpenShift Enterprise is using Ruby&Rails packaged in Software Collections.
  - OpenShift - cartridges

Tools: scl-utils

- scl-utils
  - tool for building of rpms
  - also for runtime usage
- Software Collection article on Red Hats Developers blog
- Documentation on Fedora Projects pages

Tools: scl-utils

```
[root@rhel-6-marcela ~]# scl -l
perl514
perl516
python33
ruby193
```

```
[root@rhel-6-marcela ~]# scl enable python32 bash
[root@rhel-6-marcela ~]# easy_install pip
Searching for pip
...
Installed /opt/rh/python32/root/usr/lib/python3.2/site-packages/pip-1.2.1-py3.2.egg
Processing dependencies for pip
Finished processing dependencies for pip
```


Software Collection: FHS

[root@rhel-6-marcela ~]#ls /opt/rh/ruby193/

enable root

[root@rhel-6-marcela ~]# ls /opt/rh/ruby193/root/

bin dev home lib64 mnt proc sbin srv tmp var boot etc lib media opt root selinux sys usr

Software Collection: how it works

- “rename” packages
  - rpm can't handle two versions of one software:
    - ruby193-ruby-1.9.3.327-27.srpm
    - ruby-1.9.3.327-27.srpm
- change even provides/requires of rpm

  Provides:

  ruby193-rubygem(rails) = 3.2.8

  ruby193-rubygem-rails = 1:3.2.8.el6

Software Collections: how it works

- set new paths by the 'enable' script
  - usually PATH, LD_LIBRARY_PATH, MANPATH
- ruby in collection environment

  [root@rhel-6-marcela ~]# ruby -v

  ruby 1.9.3p327 (2012-11-10 revision 37606) [x86_64-linux]
- cpan/rvm/pip will install into the collection if it's enabled

Software Collections: how it works

```
#!/usr/bin/perl

use common::sense;

say "yes";
```

```
#!/usr/bin/env perl

use common::sense;

say "yes";
```

```
#!/usr/bin/perl516-perl

use common::sense;

say "yes";
```


Software Collections: SElinux

semanage fcontext -a -e /usr /opt/rh/ruby193/root/usr

restorecon -R -v /opt/rh/ruby193/root/usr

Packaging: howto

- Documentation about Software Collections
- Fedora Project wiki SoftwareCollections

2.3. Converting a Conventional Spec File

The following steps show how to convert a conventional spec file into a Software Collection spec file so that the Software Collection spec file that you can use in both the conventional package and the Software Collection.

Procedure 2.1. Converting a Conventional Spec File into a Software Collection Spec File

1. Add the **%scl_package** macro to the spec file. Place the macro in front of the spec file preamble as follows:

```
%{?scl:%scl_package package_name}
```

2. You are advised to define the **%pkg_name** macro in the spec file in case the package is not built for the Software Collection:

```
%{!?scl:%global pkg_name %{name}}
```

   Consequently, you can use the **%pkg_name** macro to define the original name of the package wherever it is needed in the spec file that you can then use for building both the conventional package and the Software Collection.

3. Change the **Name** tag in the spec file preamble as follows:

```
Name: %{?scl_prefix}package_name
```


Software Collections: packaging

- Meta-package gives a name to the whole collection.
- Prefix of every package and provides/requires.
- All packages inside the collection must be built with enabled collection.

Tools: automatic conversion

- Specfiles can be automatically generated for packages, which are similar.
  - Perl modules, Ruby gems, Python, ...
  - cpanspec, gem2rpm, pyp2rpm, ..
- The spec2scl can transform a valid specfile into a Software Collection package.

Tools: automatic conversion

```
Requires: rubygem(actionpack) = %{version}
Requires: rubygem(actionmailer) = %{version}
Requires: rubygem(activeresource) = %{version}
Requires: rubygem(railties) = %{version}
Requires: rubygem(bundler) >= 1.0
Requires: ruby(abi) = %{rubyabi}

BuildRequires: rubygems-devel

BuildArch: noarch
Provides: rubygem(%{gem_name}) = %{version}

%description
Rails is a framework for building web-application using CGI, FCGI, mod_ruby,
or WEBrick on top of either MySQL, PostgreSQL, SQLite, DB2, SQL Server, or
Oracle with eRuby- or Builder-based templates.


%prep

%build

%install
mkdir -p %{buildroot}%{gem_dir}
gem install --local --install-dir %{buildroot}%{gem_dir} \
            --bindir %{buildroot}%{_bindir} \
            -V --no-rdoc --no-ri \
            --force %{SOURCE0}

%files
```

```
Requires: %{?scl_prefix}rubygem(actionpack) = %{version}
Requires: %{?scl_prefix}rubygem(actionmailer) = %{version}
Requires: %{?scl_prefix}rubygem(activeresource) = %{version}
Requires: %{?scl_prefix}rubygem(railties) = %{version}
Requires: %{?scl_prefix}rubygem(bundler) >= 1.0
Requires: %{?scl_prefix}ruby(abi) = %{rubyabi}

BuildRequires: %{?scl_prefix}rubygems-devel

BuildArch: noarch
Provides: %{?scl_prefix}rubygem(%{gem_name}) = %{version}

%description
Rails is a framework for building web-application using CGI,
or WEBrick on top of either MySQL, PostgreSQL, SQLite, DB2,
Oracle with eRuby- or Builder-based templates.


%prep

%build

%install
mkdir -p %{buildroot}%{gem_dir}
%{?scl:scl enable %scl "}
gem install --local --install-dir %{buildroot}%{gem_dir} \
            --bindir %{buildroot}%{_bindir} \
            -V --no-rdoc --no-ri \
            --force %{SOURCE0}
%{?scl:"}
```


Available testing repositories

Perl EL-6

- ⇨ Perl 5.16

Ruby EL-6

- ⇨ Ruby 1.9.3 + Rails 3.2.3

Python 2.7 EL-6

- ⇨ Python 2.7 + Libraries to support Django, Flask and Bottle.py (actual frameworks not included)

Python 3.3 EL-6

- ⇨ Python 3.3 + Libraries to support Django and Bottle.py (actual frameworks not included)

httpd

- ⇨ list of repositories for F-17 and EL-6

php

- ⇨ PHP 5.4 Software Collection for RHEL

mysql-5.5

- ⇨ mysql55 for RHEL-5 ⇨ mysql55 for RHEL-6 or ⇨ mysql55

postgresql-9.2

- ⇨ postgresql92 for RHEL-5 ⇨ postgresql92 for RHEL-6 or ⇨ postgresql92

unixODBC & connectors

- ⇨ unixodbc23 for RHEL-5 ⇨ unixodbc23 for RHEL-6 or ⇨ unixODBC

Thank you