RPM Packaging

How software delivery works and why RPM packaging is more current than ever

Fabio Alessandro Locati

26 October 2016

Outline

Intro

About me

- RPM user since 2001
- IT Consultant since 2004
- RPMs creator since 2013

Deployment in the past

- Mayority of application needed to be deployed on a single system
- Annual/Bi-annual
- All hands on deck
  - Devs ready to patch
  - Ops ready to deploy work-around
- Usually performed during night-time
- Hours of downtime
- Very expensive deployments

Today expectations

- Deploy multiple times every day
- Cheap deployments
- No down time
- Need of mass deployment (tens/hundreds/thousands of systems)
- Horizontal (dynamic) scalability

Deployments

Items that could be involved in deployments

- **Code**
- **Source Control System (SCM):** git, hg, svn, cvs
- **Build system:** Koji, Jenkins, Shell
- **Software packaging system:** RPM, DEB, Docker, WAR, generic archive
- **Test system:** Bodhi, Jenkins
- **Environment packaging system:** Docker, PyEnv
- **Orchestration tool:** Ansible, Puppet, Salt, Chef, Kubernetes
- **Execution environment:** Native OS, OpenStack, OpenShift, Docker, runc

RPM as software packaging system

- Advantages
  - Very well known format
  - Open Standard
  - Clear distinction between compile environment and run environment
  - Easy to integrate with any kind of environment
  - Very good at resolving dependencies
  - Checksum of all files
  - Very rigid policies
- Disadvantages
  - Heavily connected with RPM-based distro (Fedora, RHEL, OEL, SLES, OpenSUSE, CentOS, SL)
  - Very rigid policies

RPM components

- SPEC file
- Sources files (at least 1)
- Patches (eventually)

RPM Processes

RPM build process

- Fetch of the SPEC file
- Fetch of sources/patches
- Creation of the .src.rpm file
- Creation of binaries .rpm files

Fedora pipeline

- SPEC file, additional sources, and patches in git repo
- Upstream source in cache system
- Build in Koji
- Promotion to Bodhi testing branch
- Automated tests by Bodhi and AutoQA
- Manual testing
- Promotion to Bodhi stable branch

Example RPM pipeline 1

- SPEC file, additional sources, and patches in git repo
- Build in Koji
- Promotion to Bodhi testing branch
- Automated tests by Bodhi and AutoQA
- Manual testing
- Promotion to Bodhi stable branch
- Simple upgrade of live system (yum update -y PACKAGE)

Example RPM pipeline 2

- SPEC file, additional sources, and patches in git repo
- Build in Koji
- Creation of a Docker image
- Automated tests
- Manual testing
- Propagate the new Docker image

RPM and Docker

RPM and Docker

- RPM work very well in Docker environments
- Installing RPMs allow a cleaner Docker file and image
- RPMs can be deployed within or without Docker

Docker example

```
RUN dnf install -y tar make gcc ruby ruby-devel rubygems graphviz \
  rubygem-nokogiri unzip findutils which wget python-devel \
  zlib-devel libjpeg-devel redhat-rpm-config patch \
  && dnf clean packages \
  && gem install --no-ri --no-rdoc asciidoctor --version \
    $ASCIIDOCTOR_VERSION \
  && gem install --no-ri --no-rdoc asciidoctor-pdf --version \
    1.5.0.alpha.11 \
  && gem install --no-ri --no-rdoc slim \
  && (curl -LkSs https://api.github.com/repos/asciidoctor \
    | tar xfz - -C $BACKENDS --strip-components=1) \
  && wget https://bitbucket.org/pypa/setuptools/raw/bootstrap \
     -O - | python \
```

Docker example with RPM

```
RUN dnf install -y rubygem-asciidoctor-pdf \
  && dnf clean packages
```

Size of the images

- Fedora base image: 204MB
- First image: 776MB
- Second image: 238MB

Additional resources

- Laboratorio ICT, 14:00 - Come pacchettizzare applicazioni in formato RPM
- Slides: https://static.fale.io/slides/20161026-en-rpm.pdf
- Official website: http://rpm.org
- Fedora guide: https://fedoraproject.org/wiki/How_to_create_an_RPM_package
- RPM Guide: http://rpm-guide.readthedocs.io