Solving the Package Problem? Or Making it Infinitely Worse?

Joe Brockmeier
jzb@redhat.com
Twitter: @jzb
http://dissociatedpress.net/

http://projectatomic.io

Who's this guy?

Solving the Package Problem (Or Making it Infinitely Worse?)

- The Packaging Problem We Face
- Solution: Software Collections
- Solution: rpm-ostree
- Solution: Linux Containers
- Potential Pitfalls
- Questions

In the Beginning...

Distributions as the Center of the Universe

Developers do not want to be limited to system versions of software

Developers want easier ways to deploy complex software from desktop to server

Automate ALL THE THINGS

Software Collections, rpm-ostree, and Docker (oh my)

Let's talk about Software Collections

Do not require changes to RPM

Software Collections are *not* just a different version packaged for your OS

Do not overwrite system files

Example: PHP 5.4

On CentOS 6.x

PHP 5.4 package is `php54`

This pulls in:

```
php54-php-cli.x86_64
php54-php-common.x86_64
php54-php-pear.noarch
php54-php-process.x86_64
php54-php-xml.x86_64
php54-runtime.x86_64
```

Lives in: `/opt/rh/php54/root`

Avoid conflicts with system files

Require *minor* changes to your existing spec files

Do not conflict with updates on your system

Nifty: *Can* depend on other SCLs

Let's talk about using SCLs

Getting Started

- Assuming using a SLC with CentOS
  - `yum install centos-release-SCL`
  - `yum install php54` (or whatever...)
  - `scl enable php54 “application --option”`
  - Your application now uses PHP 5.4 ... the rest of the system ignores it.
  - Python & Django with SCL (by Langdon White):
    - http://red.ht/scldjango
  - Find packages for CentOS here:
    `http://mirror.metrocast.net/centos/6.5/SCL/x86_64/`

Packaging SCLs

- **Grab the necessary packages (CentOS or Fedora or RHEL 6.5):**
  `yum install scl-utils scl-utils-build`
- **Instructions on converting an existing package:** `http://bit.ly/scl-spec-file`
- **For Conversion:** `spec2scl`
- **General instructions on packaging SCLs:**
  `http://bit.ly/pkging-scls`

SoftwareCollections.org

Software Collections Currently

- See: https://www.softwarecollections.org/en/
- CentOS SCL SIG: http://wiki.centos.org/SpecialInterestGroup/SCLo
- Git repo: https://git.centos.org/project/?p=sig-sclo
- Upstream mailing list: https://www.redhat.com/mailman/listinfo/sclorg
-

What is SoftwareCollections.org?

- Upstream community for development of SCLs.
- Build and hosting services for collections.
- Resources (documentation, forums, mailing lists) for developers/packagers.
- An index of packaged software for users of CentOS, Fedora, RHEL, and other RPM-based distributions.

The Lifecycle of Collections

- SCLs can be used to provide newer software support on older releases, or (going forward) to provide legacy support on newer releases:
  - Example: Application using Ruby on Rails N deployed on CentOS 6, via SCL. Same application deployed on CentOS 7 (when released) using SCL.
- OpenShift leverages SCLs for its cartridges using RHEL supported and non-supported components.
- SCLs can be used inside Docker containers to simplify container deployment.

rpm-ostree

The Problem with Packages

- RPM (and dpkg) are designed to go one way: forward
- Upgrades are difficult to “roll back” in the event something goes wrong
- Switching between two distinct OSes / versions is more or less impossible

What is rpm-ostree?

- Derived from ostree
  - Initially conceived of as a way to parallel install multiple UNIX-like OSes (e.g., Fedora Rawhide and Fedora 20)
  - “git for operating system binaries”
- Creates an installable tree from RPMs
- Not a package manager, but does take on some of the role from package managers

What rpm-ostree Enables

- Install one or more operating system trees to a system
- Gives “atomic” updates
  - An update is, essentially, one unit – it succeeds or fails
  - An update can be rolled back
- Allows switching between “trees”
- Provides tools for creating tree composes

Current Limitations

- Currently, an rpm-ostree “tree” is an immutable system
  - Doesn't allow for adding packages to a system w/out rebuilding the tree
- Build tools are still being developed, but moving quickly

So, anybody heard of this Docker thing?

The Problem with Packages

- Deploying complex services / applications is difficult with packages
- Packages aren't as portable as we'd like
  - Application is developed on CentOS 6, but production is using CentOS 7?
- Packaging guidelines can be ... difficult
- Packages don't provide any solution for running containerized applications...

Docker: It's Like Deluxe Super Awesome Packaging

- Docker is application-centric
- Docker containers are portable
- Supports versioning for an entire container
- Components can be re-used
- Allows for supplying ready-to-run services rather than half-configured packages
- Buzzword compliant

Docker to the Rescue?

- Docker containers: relatively easy to work with
- Provide far more “services” than package systems
  - Application isolation
  - Image format, sharing, API
- Allows “layering” of applications
  - One group provides base image
  - Another group provides base image + framework/service (e.g., Apache)
  - Another group provides base image + framework + finished application ready to deploy

Pitfalls

Docker isn't Mature

- “Containers Don't Contain”
- Signing, etc. are still in their infancy
- Packaging apps in containers is still not well-understood
- Deploying apps in containers is still not well-understood

Additional Problems

- Auditing software is difficult (or impossible) in containers
- Updates to containers – who tracks? How to automate?
- Host/Container mis-matches
- What else?

Links and Pointers

- **Website:** projectatomic.io
- **Github:** github.com/projectatomic
- Facebook.com/projectatomic
- **Twitter:** @projectatomic
- **Mailing Lists:**
  http://www.projectatomic.io/community/

PROJECT ATOMIC

Thanks!

Joe Brockmeier
jzb@redhat.com
Twitter: @jzb