Continuous Packaging

Frédéric Lepied - FLOCK 2017

Who am I?

- work @ Red Hat in the leadership team of the OpenStack group
- was eNovance VP of Engineering (OpenStack service company)
- been in the Open Source world since 1995 (XFree86, Debian, MandrakeSoft/Mandriva, OpenStack)
- involved in: X11 XInput, Wacom driver, Debian, Mandrake Linux, rpmlint, msec, lads, eDeploy, OpenStack
- worked in build systems, CI, packaging for a long time

Agenda

- Introduction
- Path to Continuous Packaging
- How does it work
- Future

Introduction

Vision

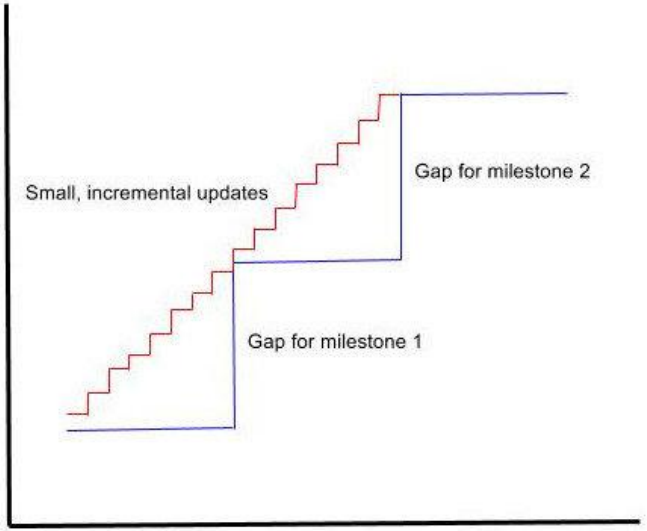
Packaging is doing integration
State of the art integration is Continuous Integration

⇒ Continuous Packaging

Traditional packaging

- Wait for a release of the upstream project to package
- Test the new package
- Report any issue upstream
- Integrate patches to fix discovered issues

⇒ It’s big bang integration!
Big batches vs small batches!



Path to Continuous Packaging

OpenStack/RDO

- Fast moving target:
  - Big number of projects: 275
  - Fast release cycle: 6 months
  - Fast moving projects: ~ 20K commits for the last cycle
  - Chicken and egg problem because the installers require up-to-date packages to work
- Solution used in RDO ⇒ DLRN:
  - Switch to a Continuous Integration mindset
  - Build a package on each upstream commit
  - Deliver a usable repository for all the built packages up to this point
  - React only on failures (email or open a gerrit review)
- https://trunk.rdoproject.org/centos7/status_report.html

Fedora

- Experimental DLRN server
- Picked 2 fast moving targets to test the value:
  a. systemd
  b. ansible
- Upstream feedback:
  a. ansible
     - Found multiple issues in unit tests and documentation build
     - Found a release issue
  b. Systemd
     - No real feedback but had to adapt to the new build system

Timeline details

http://38.145.33.116/dlrn/report.html

Delorean - Fedora-rawhide (master)

Build Date Time	Commit Date Time	Project Name	Commit Hash	Status	Repository	Build Log
2017-08-08 06:58:50	2017-08-07 15:33:59	ansible	git be5e2251a78f182374a8e40b00744b26a7905e68	SUCCESS	repo	build log
2017-08-08 05:42:30	2017-08-07 15:30:10	ansible	git bdccc2df3c2825d0e194dcd2814553ea7f9e3a98	SUCCESS	repo	build log
2017-08-08 04:26:20	2017-08-07 15:18:26	ansible	git 8f4b13de73c6889c7d0e9264404f5de940bf2f9d	SUCCESS	repo	build log
2017-08-08 03:09:58	2017-08-07 14:57:45	ansible	git b9e80b611715a739221f59ccb95a67fb09c5359a	SUCCESS	repo	build log
2017-08-08 01:53:57	2017-08-07 14:08:20	ansible	git ca7ce4459ddcf079dc6fc1112d6e3fdd7316a105	SUCCESS	repo	build log
2017-08-08 00:37:24	2017-08-07 13:58:47	ansible	git f1816bb438819e0dc56441e96f365b065bd08633	SUCCESS	repo	build log
2017-08-07 23:21:37	2017-08-07 13:53:20	ansible	git ac15df8ab9c1937553894f26b433a24447a0fc6f	SUCCESS	repo	build log
2017-08-07 22:05:19	2017-08-07 12:36:51	ansible	git 2276c5f2060a46d556268b846e6ef30950d992cc	SUCCESS	repo	build log
2017-08-07 20:47:20	2017-08-07 11:19:31	ansible	git 2d5908255a26d90b4ddec1235f064662462c0dd1	SUCCESS	repo	build log
2017-08-07 20:44:59	2017-08-07 09:29:20	systemd	git c23c34bcba944d597bdc8e210fb9f0e00d662629	FAILED	repo	build log
2017-08-07 20:42:37	2017-08-07 09:27:24	systemd	git 15d167f8a30597cb93996e5608adb496507ab9d7	FAILED	repo	build log
2017-08-07 20:40:16	2017-08-07 09:06:07	systemd	git ca992ecf073cc3eedc40169d0f2b23c990807d0f	FAILED	repo	build log
2017-08-07 20:37:16	2017-08-07 09:02:47	systemd	git 2d35b79cdc65952a71b768f4fc0e2134a47e0294	FAILED	repo	build log
2017-08-07 20:34:16	2017-08-07 08:52:27	systemd	git b5338ddcfd631cf1f30910ef70f1c5b3566f082d	FAILED	repo	build log
2017-08-07 19:17:45	2017-08-07 08:48:30	ansible	git fa6ce54011ffd7974d5969fb39d033c3f3d72a89	SUCCESS	repo	build log

Summary view

http://38.145.33.116/dlrn/status_report.html

Delorean - Fedora-rawhide (master)

Project Name	Status	First failure after success	Number of days since last success
ansible	SUCCESS		
systemd	FAILED	git 785889e56d26c60755c4a76b1dc5eccd8296a6df (build log)	10 days

How does it work

How does it work?

Configuration files:

- INI file to describe the DLRN server settings
- mock config file
- YAML file to describe the git repositories

Steps:

- DLRN fetches git repositories for upstream and dist-git
- DLRN builds a package using mock on every commit
- On successful build and install a repository is created with all the packages included
- Repositories are kept to allow to do bisections and comparisons with past versions

INI file to configure DLRN

```
datadir=./rawhide-data
scriptsdir=./scripts
baseurl=http://38.145.33.116/dlrn/
distro=master
source=master
target=fedora-rawhide
smtpserver=localhost
reponame=delorean
templatedir=./dlrn/templates
maxretries=3
database_connection=sqlite:///rawhide.sqlite
fallback_to_master=1
```

Mock config file

```
config_opts['root'] = 'dlrn-fedora-x86_64'
config_opts['target_arch'] = 'x86_64'
config_opts['legal_host_arches'] = ('x86_64',)
config_opts['chroot_setup_cmd'] = 'install basesystem
rpm-build python2 git'
config_opts['dist'] = 'rawhide'
config_opts['extra_chroot_dirs'] = [ '/run/lock', ]
config_opts['releasever'] = '27'
...
```

YAML file to describe the projects and packages

```
releases:
- name: rawhide
  branch: master
  repos:
  - name: fc27
    distrepos:
    - name: Fedora Rawhide
      url: https://mirrors.fedoraproject.org/metalink?repo=fedora-rawhide&arch=x86_64
package-configs:
  pkgconf:
    maintainers:
    - flepied@redhat.com
    - yguenane@redhat.com
package-default:
  name: "%(project)s"
  upstream: https://github.com/%(project)s/%(project)s
  master-distgit: file:///home/fedora/git/%(project)s
packages:
- project: systemd
  conf: pkgconf
  distro-branch: rpm-master
- project: ansible
  conf: pkgconf
  distro-branch: rpm-devel
  source-branch: devel
```

Packaging flow in DLRN

For each project to build:

1. Extract the upstream repo under <datadir>/<project>
2. Extract the dist-git repo under <datadir>/<project>_distro
3. Switch to the git target commits for upstream and dist-git
4. Create a tar ball like it would be done for a release
5. Subtitute tar ball name and version into the spec file
6. Create an src.rpm
7. Build and install using mock
8. On success create the metadata for the rpm repository
9. On the first failure send an email to the packagers

Organization on disk

```
<datadir>/<project>/
<datadir>/<project>_distro/
<datadir>/repos/<1-2>/<3-4>/<sha1>/rpmbuild.log
<datadir>/repos/<1-2>/<3-4>/<sha1>/*.rpm
<datadir>/repos/<1-2>/<3-4>/<sha1>/repodata/
```

Optimization

If your server cannot keep up with the changes, you can build only the head of the branch on each project (--head-only option).

Then in case of an error, to detect the commit that caused it just use the git bisect helper:

```
./scripts/bisect.sh <conf file> <package> <good sha1> <bad sha1>
```

Future

Next steps

- Gather feedback from the Fedora community
- Put an official server in place
- Involve the official packagers and add new packages
- Use the dist-git pagure server to host the development branches to allow easy collaboration
- Create a plugin for modulemd files
- Connect the Fedora CI system to consume these repositories

Even shorter feedback loop

- Build packages on PR and test them

2 ways to do it:

1. Add a webhook to build on each proposed change
2. Process all available proposed changes in batch mode filtering them to diminish the number

Conclusion

Summary

Continuous packaging

- aligns development and packaging cycles
- reduces the number of patches to carry
- spreads the load to fix issues
- reduces time to have a new version of a package in Fedora

Call for action

Let’s put Continuous Packaging in place in Fedora!

Questions?

Contact

flepied@redhat.com
IRC: flepied