packstation

A Git/Gitlab-driven workflow for internal package repositories and PPAs

Hilko Bengen
bengen@debian.org

2019-06-08

whoami

- Debian user at home since 1996
- Using Debian in day jobs since 2000
- DD since 2003
- Maintainer of probably too many packages

At work...

- DCSO, managed (IT-)Security Service (MSS) Provider
- Big user of Debian & Ubuntu, for server and workstation environments
- Usually stable + security updates + selected backports
- Great base to work with and build upon

However…

- What about locally developed software?
- What about other backports?
- What about local forks, bugfixes?
- No YOLO Ansible `/usr/local` installations or lengthy Wiki installation instructions, please!
- Work **with** the distribution's infrastructure, not **against** it

User's Point Of View

- Use `ftp.*.{debian.org,ubuntu.com}` for base install
- Add local repositories, specific to my distribution
- Somebody^{TM} keeps the packages I need up-to-date, so I don't have to care.

Developer's/Packager's Point Of View

- git, Gitlab
- Configure a simple webhook for tag push events: `http://packstation.example.org/gitlab-webhook`
  - Remote URLs for `gitlab-ci.yml` files instead would have been great
- Use *git-buildpackage(1)* baed workflow, including *pristine-tar(1)*
  - Really looking forward to integrating *pristine-lfs*
- Automated build is triggered by magic git tags
  - Just like `upstream/$VERSION`, `debian/$VERSION`
- Feedback about success/failure is provided through Gitlab's comment function.

Components

- Gitlab's webhook feature
- *nginx* (environment for the webhook receiver)
- *git-buildpackage(1)*
- *dput(1)*
- A single *reprepro(1)* instance
- *sbuild(1)*
- A set of rather simple scripts
  - held together by simple filesystem-based message queues and *inotify(7)*

From git to source package

- changelog entry

```
hello (2.10-2~local+1) stretch xenial; urgency=medium

  * Trivial backport for demo purposes

 -- Hilko Bengen <bengen@debian.org>  Sun, 02 Jun 2019 09:31:58 +0200
```

- `git tag $MAGIC/2.10.2_local+1 && git push && git push --tags`
- Webhook receiver
- *packstation-build-dsc*
  - Checks PGP signature on magic tag
  - `gbp buildpackege -S [...]  --git-export=refs/tag/$TAG`
  - Adds custom headers with context for later steps: Commit ID, project ID, etc.
  - Rewrites `Distributions` field in `.changes` file, e.g.:
    - `stable` → `stretch`, `xenial`, `bionic`
    - `debian` → `stretch`, `buster`
    - `ubuntu` → `xenial`, `bionic`
  - `dput`

From source package to binary packages

- *packstation-reprepro-handle-dsc*
  - Handles log events generated by *reprepro* on source package import
  - Schedules autobuilder jobs for each supported architecture/distribution combination
- *packstation-build-deb*
  - Runs `sbuild`
  - Adds distribution-specific build version number suffix, e.g.:
    - stretch → `+deb9`
    - buster → `+deb10`
    - xenial → `+ubu1604`
    - bionic → `+ubu1804`
  - `dput`

What about PPAs, then?

- Vaporware; sorry, I am still working on this
- PPA-specific webhook URL
  `http://packstation.example.org/gitlab-webhook?target=$USER/$TOPIC`
- Management interface for users, associated PGP keys, Gitlab API keys, magic tag prefix, etc.
- Auto-generate configuration for multiple *reprepro* repositories
- Separate repositories for testing/staging
- API for package migration/removal
  - No *dcut(1)*-style user interface, please
- Make it possible to run binary build jobs on separate machines

Questions?

©2004 KMJ https://de.wikipedia.org/wiki/Datei:Packstation_01_KMJ.jpg