Bits from the Stable Release Management

Debconf 10

Philipp Kern

August 7, 2010

Table of contents

What does Stable Release Management do?

- accept security advisories into proposed-updates
- review other fixes targetted to "stable" (mainly those sent for approval to `debian-release@lists.debian.org` and those in the p-u-NEW queue)
- prepare a point release announcement
- coordinate with stable kernel management and debian-installer team
- coordinate with FTP masters, press, security and CD team to set a suitable point release date
- ensure architectures being in sync, no missing builds
- (likewise for oldstable)

“proposed-updates”

- “proposed-updates” suite is moderated through proposed-updates-NEW
- packages only enter proposed-updates when the RMs tell dak to accept it
- packages in proposed-updates are supposed to work, every bug encountered should be reported to us ASAP

Tool: Queue Overview

- “NEW queue summary for stable-proposed-updates” as linked from `http://release.debian.org`
- automatic debdiff against the current version (i.e. either stable or proposed-updates)
- automated checks for version and installability problems
- tracking missing builds, out-of-dates
- helping to mostly autogenerate point release updates
- also listing point release TODO items and removal requests

Open questions

- Who is using “proposed-updates”?
- How to collect test reports?
- This might require checklists for test reports?

Point Release Intervals

- We aim for:
  - two months between point releases in the “stable” timeframe
  - four months or more if it’s “oldstable”
- some skew might be introduced by ensuring that two do not collide

Updates besides Point Releases

- some packages require timely updates besides normal point releases
- e.g. tzdata ships newer timezone definitions that should be pushed to the users, but using security is not appropriate
- e.g. clamav might need updates to cope with new virus signatures
- e.g. pidgin/libpurple might need protocol updates to keep up with non-free IM services

Flash back: volatile

- volatile was introduced with two suites:
    - volatile proper: a set of packages everyone can update to (historically: clamav-data, tzdata, etc.)
    - volatile-sloppy: packages that need larger version bumps to get useful again (historically: gaim)
- separate team, separate infrastructure (own archive host)

Problems with volatile at the moment

- run basically by one person
- ancient dak version (with a version still maintained in bzr)
- no support for source version 3, etc.
- no ability to copy over volatile builds and sources to proposed-updates
- *at least:* mirroring now handled by `mirroradm`

Proposal: integration into SRM

- run volatile on the normal infrastructure
- use volatile as a suite to pass updates more quickly to the users than point releases can (c.f. -updates on Ubuntu)
- copy into volatile from proposed-updates
- copy volatile bits into stable proper at point release time
- **goal:** keep stable as usable as possible

Policies

The old update policy

1. The package fixes a security problem. An advisory by our own Security Team is required. Updates need to be approved by the Security Team.

The old update policy

1. The package fixes a security problem. An advisory by our own Security Team is required. Updates need to be approved by the Security Team.
2. The package fixes a critical bug which can lead to data loss, data corruption, or an overly broken system, or the package is broken or not usable (anymore).

The old update policy

1. The package fixes a security problem. An advisory by our own Security Team is required. Updates need to be approved by the Security Team.
2. The package fixes a critical bug which can lead to data loss, data corruption, or an overly broken system, or the package is broken or not usable (anymore).
3. The stable version of the package is not installable at all due to broken or unmet dependencies or broken installation scripts.

The old update policy

1. The package fixes a security problem. An advisory by our own Security Team is required. Updates need to be approved by the Security Team.
2. The package fixes a critical bug which can lead to data loss, data corruption, or an overly broken system, or the package is broken or not usable (anymore).
3. The stable version of the package is not installable at all due to broken or unmet dependencies or broken installation scripts.
4. All released architectures have to be in sync.

The old update policy

1. The package fixes a security problem. An advisory by our own Security Team is required. Updates need to be approved by the Security Team.
2. The package fixes a critical bug which can lead to data loss, data corruption, or an overly broken system, or the package is broken or not usable (anymore).
3. The stable version of the package is not installable at all due to broken or unmet dependencies or broken installation scripts.
4. All released architectures have to be in sync.
5. The package gets all released architectures back in sync.

The old update policy

1. The package fixes a security problem. An advisory by our own Security Team is required. Updates need to be approved by the Security Team.
2. The package fixes a critical bug which can lead to data loss, data corruption, or an overly broken system, or the package is broken or not usable (anymore).
3. The stable version of the package is not installable at all due to broken or unmet dependencies or broken installation scripts.
4. All released architectures have to be in sync.
5. The package gets all released architectures back in sync.

- It is "(or (and (or 1 2 3) 4) 5)" (yay, LISP syntax!).

The old update policy (2)

- Regular bugs and upgrade problems don't get fixed in new revisions for the stable distribution.
- Packages which will most probably be rejected:
    - Packages that fix non-critical bugs
    - Packages for which its binary packages are out of sync with regard to all supported architectures in the stable distribution
    - Binary packages for which the source got lost somehow
    - Packages that fix an unusable minor part of a package

The “new“ update policy

- Security advisories go all in, if they have the necessary builds in the `proposed-updates` suite.
- A patch that was not previously acceptable by the old rules is likely to be acceptable for a stable update if it:
  - fix a security issue, or
  - fix a bug of at least severity important, or
  - fix an installability (binNMU), an FTBFS bug, or
  - bring architectures back in sync
- Common sense with individual updates handled on a case-by-case basis
- Every update risks regressions (e.g. by being rebuilt), we need to minimize this impact
- If you think that stable packages should get fixed in some regards, please do not hesitate to contact us!

Packages that will get newer upstream versions

- PostgreSQL (new bugfix releases)
- clamav (bugfix and security releases)
- tzdata (timezone data)
- Mozilla-related packages (Iceweasel, Icedove)
- **problem:** how to tag packages of which we know that they get larger updates

Questions? Comments?