What git-buildpackage (hopefully) can do for you.

Maintaining Debian packages in Git

Guido Günther *<agx@sigxcpu.org>*

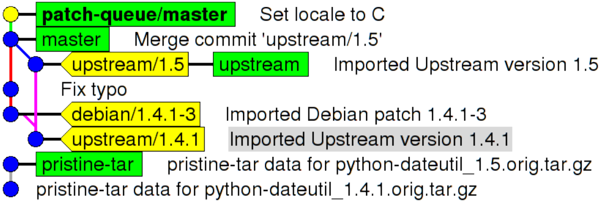
Debconf 10, NYC

Overview

1. Introduction
2. Importing, updating and building packages
3. Changelog and commits
4. More tools
5. Where to go from here?

Git distributed version control system

- Stores different source code revisions (trees)
- and the relationship among them (commits)
- You can branch to work on different topics...
- ...and merge back the changes
- You can mark (tag) a source code revision
- All actions are on the local repository
- You can publish the whole repository or certain branches
- space efficient, fast



patch-queue/master Set locale to C
master Merge commit 'upstream/1.5'
upstream/1.5 upstream Imported Upstream version 1.5
Fix typo
debian/1.4.1-3 Imported Debian patch 1.4.1-3
upstream/1.4.1 Imported Upstream version 1.4.1
pristine-tar pristine-tar data for python-dateutil_1.5.orig.tar.gz
pristine-tar data for python-dateutil_1.4.1.orig.tar.gz

Recurring tasks with Debian source packages

- Import a Debian package - for nmu, qa or sponsoring
- Change, build, test, change, build, test, ...
- Import a new upstream version
- Add, remove and update patches
- Write debian/changelog
- Build backports, stable updates, etc.

Importing a Debian package

- Import source package from the local file system:

```
git-import-dsc python-dateutil_1.4.1-1.dsc
```

- Import source package from an URL:

```
git-import-dsc --download http://....dsc
```

- Import source package via apt-get:

```
git-import-dsc --download python-dateutil
```

Importing a Debian package

- Import source package from the local file system:

```
git-import-dsc python-dateutil_1.4.1-1.dsc
```

- Import source package from an URL:

```
git-import-dsc --download http://....dsc
```

- Import source package via apt-get:

```
git-import-dsc --download python-dateutil
```

- This also works incrementally.

Importing a Debian package II

- Import multiple versions from the local file system:

```
git-import-dscs *.dsc
```

- Import source packages from snapshots.debian.org:

```
git-import-dscs --debsnap python-dateutil
```

Modifying a Debian package

Important Git commands:

- Did something change:
  git status, *git diff*
- Add and remove files:
  git add, *git rm*
- Record changes:
  git commit
- Look at the history of changes:
  git log, *git whatchanged*
- Correct mistakes:
  git checkout, *git reset*
- Generate a patch:
  git format-patch → reportbug

Building and testing

- Write the changelog

  ```
  git-dch --snapshot --auto
  ```

Building and testing

- Write the changelog

  ```
  git-dch --snapshot --auto
  ```

- Build the package

  ```
  git-buildpackage --git-ignore-new
  ```

Building and testing

- Write the changelog

  ```
  git-dch --snapshot --auto
  ```

- Build the package

  ```
  git-buildpackage --git-ignore-new
  ```

- Everything o.k.? Then:

  ```
  git-dch --release -a; git-buildpackage --git-tag
  ```

Importing a new upstream version

- Import tarball form local file system:

  ```
  git-import-orig python-dateutil_1.5.orig.tar.gz
  ```

- Import from upstream URL:

  ```
  git-import-orig --uscan
  ```

Upstream sources in Git

Why keep upstream sources in Git?

- You can apply patches directly and test them
- You can easily revert changes and experiment
- It's easy to compare versions and spot changes

```
git diff upstream/1.0..upstream/2.0 [file|directory]
```

- Upstream might be using Git

Quilt patches

- Patches: *debian/patches/*.{patch,diff}*
- Apply: debian/patches/series
- dh –with quilt, patchsys-quilt, source format 3.0 (quilt)

Quilt patches

- Patches: *debian/patches/*.{patch,diff}*
- Apply: debian/patches/series
- dh –with quilt, patchsys-quilt, source format 3.0 (quilt)

- Import patches:

  ```
  gbp-pq import
  ```
- Export patches:

  ```
  git checkout master; gbp-pq export
  ```
- Update patches:

  ```
  gbp-pq rebase
  ```
- Drop patch-queue branch:

  ```
  gbp-pq drop
  ```

Quilt patches

- Patches: *debian/patches/*.{patch,diff}*
- Apply: debian/patches/series
- dh –with quilt, patchsys-quilt, source format 3.0 (quilt)

- Import patches:

  ```
  gbp-pq import
  ```
- Export patches:

  ```
  git checkout master; gbp-pq export
  ```
- Update patches:

  ```
  gbp-pq rebase
  ```
- Drop patch-queue branch:

  ```
  gbp-pq drop
  ```

Simple patch system - alternatives: TopGit, TNT

Source format 3.0 (quilt)

- dpkg-source –unapply-patches (debian/source/local-options)
- gbp-pq import
- git-buildpackage –git-export-dir=../build-area

More on git-dch

Commit-Message:

```
commit a7fe7c4678a48072c11bb57fd1f99ca7b8118158
Author: Guido Günther <agx@sigxcpu.org>
Date:   Thu Jan 28 19:03:33 2010 +0100

    Add basic bash completion for git-buildpackage

    Thanks: Siegfried-Angel Gevatter
    Closes: #567313
```

debian/changelog:

```
  * [a7fe7c4] Add basic bash completion for git-buildpackage (Closes: #567313)
    - thanks to Siegfried-Angel Gevatter
```

More on git-dch

Commit-Message:

```
commit a7fe7c4678a48072c11bb57fd1f99ca7b8118158
Author: Guido Günther <agx@sigxcpu.org>
Date:   Thu Jan 28 19:03:33 2010 +0100

    Add basic bash completion for git-buildpackage

    Thanks: Siegfried-Angel Gevatter
    Closes: #567313
```

debian/changelog:

```
  * [a7fe7c4] Add basic bash completion for git-buildpackage (Closes: #567313)
    - thanks to Siegfried-Angel Gevatter
```

- Tags: *Closes:*, *Thanks:*
- –id-length

cl2vcs

- –id-length: Changelog entry ↔ Commit
- via the Vcs-Git: link we can find the Git repo

cl2vcs

- –id-length: Changelog entry ↔ Commit
- via the Vcs-Git: link we can find the Git repo
- → So you can browse commits without downloading the package:

`https://honk.sigxcpu.org/cl2vcs/?pkg=libvirt`

Configuration

- */etc/git-buildpackage/gbp.conf*
- ˜/.gbp.conf
- <repo>/.git/gbp.conf
- <repo>/debian/gbp.conf

Configuration

- */etc/git-buildpackage/gbp.conf*
- ˜/.gbp.conf
- <repo>/.git/gbp.conf
- <repo>/debian/gbp.conf

- Tag- and branch naming patterns
- Sign tags
- Use pristine-tar
- Build commands (e.g. using pbuilder)
- Export source tree (svn-buildpackage like)
- Tarball compression
- Import filters
- Hooks (prebuild, postbuild, posttag)

Configuration

- */etc/git-buildpackage/gbp.conf*
- ˜/.gbp.conf
- <repo>/.git/gbp.conf
- <repo>/debian/gbp.conf

- Tag- and branch naming patterns
- Sign tags
- Use pristine-tar
- Build commands (e.g. using pbuilder)
- Export source tree (svn-buildpackage like)
- Tarball compression
- Import filters
- Hooks (prebuild, postbuild, posttag)

```
git-* --help
```

shows the current defaults.

More tools around GBP...

- Remote repositories:
  - gbp-clone: Clone from remote setting up all branches
  - gbp-pull: Update branches from remote
  - /u/s/d/gbp/examples/gbp-posttag-push
  - /u/s/d/gbp/examples/gbp-create-remote-repo

More tools around GBP...

- Remote repositories:
  - gbp-clone: Clone from remote setting up all branches
  - gbp-pull: Update branches from remote
  - /u/s/d/gbp/examples/gbp-posttag-push
  - /u/s/d/gbp/examples/gbp-create-remote-repo
- pristine-tar
- git-pbuilder: `http://www.eyrie.org/~eagle/software/scripts/git-pbuilder.html`
  Configuration: `http://www.eyrie.org/~eagle/software/scripts/git-pbuilder.html`
- git hooks: `http://git.debian.org/?p=users/seanius/vcs-hooks/git-hooks.git`
- `http://git.debian.org/`

Where to go from here?

- Multiple tarball support
- git-dch: Make better use of dch (–nmu, –bpo)
- Improve support for complex patch sets (TopGit, etc.)

git-buildpackage

- Wiki: https://honk.sigxcpu.org/piki/projects/git-buildpackage/
- Manual: http://honk.sigxcpu.org/projects/git-buildpackage/manual-html/gbp.html

git-buildpackage

- Wiki: `https://honk.sigxcpu.org/piki/projects/git-buildpackage/`
- Manual: `http://honk.sigxcpu.org/projects/git-buildpackage/manual-html/gbp.html`

- Thanks: git, devscripts, pristine-tar, bug reporters (especially the pkg-ocaml maintainers)
- Questions? Comments?

References

References to topics that came up during the talk:

- Migration from svn: `http://wiki.debian.org/Teams/OCamlTaskForce/GitMigration`
- git-pbuilder configuration: `http://www.eyrie.org/~eagle/software/scripts/git-pbuilder.html#S2`

gbp-pq example

- Get the example:

```
git-import-dsc libvirt
cd libvirt
```

- Create the patch-queue branch:

```
gbp-pq import
```

- Make any changes and commit them
- Serialize the patch-queue:

```
gbp-pq export
```

- You should now see the commits from above as patches in debian/patches
- Commit the new patches:

```
git commit -m"Commit message" debian/patches
```

- Alternatively gbp-add-patch will fetch the commit message from the patch description and commit:

```
/usr/share/doc/git-buildpackage/examples/gbp-add-patch \
                          debian/patches/0008-your-new.patch
```

- Update the changelog and drop patch-queue branch

```
git-dch -S -a
gbp-pq drop
```

More on configuration

~/.gbp.conf:

```
[DEFAULT]
# Use pristine-tar everywhere:
pristine-tar = True

[git-buildpackage]
# Automatically push to remote repo after tagging a new release
posttag = /usr/share/doc/git-buildpackage/examples/gbp-posttag-push
# Run Lintian after a succesful build
postbuild = lintian $GBP_CHANGES_FILE

[git-dch]
# Parse meta tags like Closes: from commit messages
meta = True
# Add seven digits of the commits sha1 to the commit message
id-length = 7
# Regex for matching bugs to close
meta-closes = Closes|LP
# Use the full commit message instead of the first line only
full = True
# Ignore these in commit messages
ignore-regex = (Signed-off|Acked)-by:
# Use git committer as author
git-author = True
```

Creating remote repos

- Create local repository
- Package and commit initial version
- Check remote repo location

  ```
  gbp-create-remote-repo --help
  ```
- Push changes into remote repo

  ```
  gbp-create-remote-repo --remote-url-pattern=ssh://srv/dir/%(pkg)s.git
  ```
- Update .git/config as needed

What does the Git repository look like?

Commits, Tags, Branches

```
* 23667ff (HEAD, master) Bump standards version
*   eef58a9 Merge commit ’upstream/1.5’
|\
| * 46ca33a (tag: upstream/1.5, upstream) Imported Upstream version 1.5
* | 23396fc Document changes and release 1.4.1-4
* | e181ff2 Remove comment
* | 539f202 (tag: debian/1.4.1-3) Imported Debian patch 1.4.1-3
|/
* 4e5dda0 (tag: upstream/1.4.1) Imported Upstream version 1.4.1
```

- Upstream branch: upstream

What does the Git repository look like?

Commits, Tags, Branches

```
* 23667ff (HEAD, master) Bump standards version
*   eef58a9 Merge commit ’upstream/1.5’
|\
| * 46ca33a (tag: upstream/1.5, upstream) Imported Upstream version 1.5
* | 23396fc Document changes and release 1.4.1-4
* | e181ff2 Remove comment
* | 539f202 (tag: debian/1.4.1-3) Imported Debian patch 1.4.1-3
|/
* 4e5dda0 (tag: upstream/1.4.1) Imported Upstream version 1.4.1
```

- Upstream branch: upstream
- Debian branch: master

What does the Git repository look like?

Commits, Tags, Branches

```
* 23667ff (HEAD, master) Bump standards version
*   eef58a9 Merge commit ’upstream/1.5’
|\
| * 46ca33a (tag: upstream/1.5, upstream) Imported Upstream version 1.5
* | 23396fc Document changes and release 1.4.1-4
* | e181ff2 Remove comment
* | 539f202 (tag: debian/1.4.1-3) Imported Debian patch 1.4.1-3
|/
* 4e5dda0 (tag: upstream/1.4.1) Imported Upstream version 1.4.1
```

- Upstream branch: upstream
- Debian branch: master
- Upstream tag: upstream/<version>

What does the Git repository look like?

Commits, Tags, Branches

```
* 23667ff (HEAD, master) Bump standards version
*   eef58a9 Merge commit ’upstream/1.5’
|\
| * 46ca33a (tag: upstream/1.5, upstream) Imported Upstream version 1.5
* | 23396fc Document changes and release 1.4.1-4
* | e181ff2 Remove comment
* | 539f202 (tag: debian/1.4.1-3) Imported Debian patch 1.4.1-3
|/
* 4e5dda0 (tag: upstream/1.4.1) Imported Upstream version 1.4.1
```

- Upstream branch: upstream
- Debian branch: master
- Upstream tag: upstream/<version>
- Debian tag: debian/<debian-version>

What does the Git repository look like?

Commits, Tags, Branches

```
* 23667ff (HEAD, master) Bump standards version
*   eef58a9 Merge commit 'upstream/1.5'
|\
| * 46ca33a (tag: upstream/1.5, upstream) Imported Upstream version 1.5
* | 23396fc Document changes and release 1.4.1-4
* | e181ff2 Remove comment
* | 539f202 (tag: debian/1.4.1-3) Imported Debian patch 1.4.1-3
|/
* 4e5dda0 (tag: upstream/1.4.1) Imported Upstream version 1.4.1
```

- Upstream branch: upstream
- Debian branch: master
- Upstream tag: upstream/<version>
- Debian tag: debian/<debian-version>
- These are the defaults.

What does the Git repository look like?

Commits, Tags, Branches

```
* 23667ff (HEAD, master) Bump standards version
*   eef58a9 Merge commit ’upstream/1.5’
|\
| * 46ca33a (tag: upstream/1.5, upstream) Imported Upstream version 1.5
* | 23396fc Document changes and release 1.4.1-4
* | e181ff2 Remove comment
* | 539f202 (tag: debian/1.4.1-3) Imported Debian patch 1.4.1-3
|/
* 4e5dda0 (tag: upstream/1.4.1) Imported Upstream version 1.4.1
```

- Upstream branch: upstream
- Debian branch: master
- Upstream tag: upstream/<version>
- Debian tag: debian/<debian-version>
- These are the defaults.
- Patches: patch-queue/master

What does the Git repository look like?

Commits, Tags, Branches

```
* 23667ff (HEAD, master) Bump standards version
*   eef58a9 Merge commit ’upstream/1.5’
|\
| * 46ca33a (tag: upstream/1.5, upstream) Imported Upstream version 1.5
* | 23396fc Document changes and release 1.4.1-4
* | e181ff2 Remove comment
* | 539f202 (tag: debian/1.4.1-3) Imported Debian patch 1.4.1-3
|/
* 4e5dda0 (tag: upstream/1.4.1) Imported Upstream version 1.4.1
```

- Upstream branch: upstream
- Debian branch: master
- Upstream tag: upstream/<version>
- Debian tag: debian/<debian-version>
- These are the defaults.
- Patches: patch-queue/master
- Binary delta: pristine-tar