Maintenir son paquet avec git-buildpackage

- Objet : Installation, configuration et utilisation de git-buildpackage
- Niveau requis :
  avisé
- Commentaires : *Pour maintenir vos paquets Debian sous git avec git-buildpackage.*
- À savoir : Contribuer à Debian
- Suivi :
  à-tester
  - Création par captnfab 28/02/2014
  - Testé par <...> le <...>
- Commentaires sur le forum : ici [1)]

Nota :

Contributeurs, les Fix Me! sont là pour vous aider, supprimez-les une fois le problème corrigé ou le champ rempli !

Introduction

Les paquets évoluent, au fil des versions, les patchs à effectuer pour intégrer correctement les paquets à Debian diffèrent. Les corrections de bugs sont intégrées par *upstream*[2)]. Pour s'y retrouver, il est conseillé de maintenir les sources de son paquet Debian via un VCS[3)], tel que **subversion**, **mercurial** ou **git**.

Nous allons traiter ici le cas d'un paquet géré sous **git** via les utilitaires **git-buildpackage**.

Installation

On installe *git-buildpackage* ainsi que quelques outils qui nous seront utiles :

```
apt-get install git gitk git-buildpackage pristine-tar
```

Configuration

Configuration de Git

```
git config --global user.name "Votre Nom"
```

```
git config --global user.email votre@adresse-email.org
```

Si vous avez une clé GPG Fix Me! :

```
git config --global user.signingkey "0xXXXXXXXX"
```

Configuration de Git-BuildPackage

Créer le fichier `~/.gbp.conf` contenant le texte suivant :

~/.gbp.conf

```
[DEFAULT]
# Sign tags by default:
sign-tags = True
keyid = 0xXXXXXXXX
# use pristine tar by defaul
pristine-tar = True

[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|BZ
# Use the full commit message instead of the first line only
full = True
# Ignore these in commit messages
ignore-regex = (Signed-off|Acked)-by:

[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
# Build command
builder = dpkg-buildpackage -i -I -uc -us
# Clean command
cleaner = /bin/true

[import-orig]
# Automatically forward the changelog after importing a new upstream
version
postimport = git-dch -N%(version)s -S -a --debian-branch=$GBP_BRANCH
```

Remplacer 0xXXXXXXXX par l'id de votre clé GPG !

Utilisations

Gérer les patchs via une branche ''patch-queue''

Pour gérer les patchs de debian/patchs, on peut utiliser une branche git.

- Cela nous permet pour chaque branche *foo* d'avoir la branche non-patchée (*foo*) et la branche patchée (*patch-queue/foo*)
- Un commit dans la *patch-queue* correspond exactement à un patch dans `debian/patches/`
- Les patchs peuvent facilement être supprimés, ou ajoutés en modifiant la branche *patch-queue* (et en évitant de se tromper en utilisant *quilt add*, *dpatch-edit-patch*, *cdbs-edit-patch*, etc.)
- Les patchs peuvent facilement être portés sur les nouvelles versions grâce à `git rebase` sur la branche *patch-queue* (les patchs appliqués par *upstream* sont automatiquement détectés).
- Les patchs générés dans `debian/patches/` ont toutes les informations nécessaires pour être envoyés à *upstream*.

Inconvénient :

- Pas d'historique sur la branche *patch-queue/foo*, mais bien sûr l'historique de la branche *foo* reste disponible.

Créer la branche de patchs

```
gbp pq import
```

Cela changera la branche courante du dépôt à *patch-queue/master*.

Travailler sur les patchs

- Un commit = un patch
- C'est une branche git classique

Re-générer les fichiers de patchs

```
gbp pq export
```

Cela changera la branche courante du dépôt à *master*.

Enregistrer les modifications

```
git add debian/patches
```

```
git commit
```

Mettre à jour le changelog

```
git-dch -S -a
```

Construire le paquet

- Voir plus bas.

Après avoir importé une nouvelle release

```
  gbp pq rebase
git checkout master
gbp pq export
```

Si des commits ont été fait sur //master// depuis la dernière mise à jour

```
gbp pq rebase
```

Créer un premier patch

- `gbp pq import`
- Créer le patch et en faire un commit
- Générer le fichier patch pour quilt :

```
git checkout master
gbp pq export
```

- Commit:

```
git add -a debian/patches/
```

```
git commit -m 'mon premier patch'
```

Cloner un dépôt git-buildpackage existant

```
gbp clone adresse-du-dépôt-git
```

Garder un dépôt jour

Après un clone initial avec `gbp clone`, vous pouvez exécuter *gbp pull* pour mettre à jour les branches *debian*, *upstream* et *pristine-tar* depuis le dépôt distant. Le flot de travail ressemble alors à cela :

- Initiallement, cloner le dépôt une fois

```
gbp clone adresse/du/dépôt.git
cd dépôt
```

- Travailler sur ce clone, faire des commits, récupérer les releases, faire des pushs, etc.
- Pour récupérer après quelques jours les mises à jours faites par vos collègues :

```
gbp pull --redo-pq
```

Cela mettra à jour toutes les branches en fonction des modifications faites par vos collaborateurs, et regénérera la branche de patchs.

Compilation du paquet

À faire depuis la branche *patch-queue/master* à jour :

```
export DEB_HOST_ARCH=amd64
export ARCH=amd64
export DIST=sid
git-buildpackage --git-pbuilder --git-arch=$ARCH --git-dist=$DIST --git-
debian-branch=patch-queue/master
```

Vous pouvez ajouter l'option `--git-tag` pour tagger et uploader la version en cours.

Quelques rappels de git

- Supprimer un tag localement et à distance :

```
git tag -d nomdutag && git push origin nomdutag
```

- Revenir irréversiblement à un ancien commit :

```
git reset --hard hashprefix-du-commit
```

- Établir une branche basée sur un ancien commit :

```
git checkout -b nomdelabranche hashprefix-du-commit
```

Références

- How do you remove a tag from a remote repository
- How to revert to a previous git commit

Références

- Debian Mentors
- Debian packages in git
- Building Debian Packages with git-buildpackage
- Utiliser Git sur Alioth

1)

N'hésitez pas à y faire part de vos remarques, succès, améliorations ou échecs !

2)

le développeur amont du logiciel empaqueté

3)

Gestionnaire de versions concurentes