Building Debian Packages with hg-buildpackage

John Goerzen

Building Debian Packages with hg-buildpackage

by John Goerzen

Table of Contents

1. Introduction.....1

Purpose.....1

Features.....1

Usage Overview.....2

2. Getting Started.....3

The hg-buildpackage Repository Layout.....3

Upstream Repository.....3

Debian Repository.....3

3. Importing Packages.....4

Importing Debian Source Packages.....4

Importing Upstream Sources.....4

4. Common Tasks.....6

Building and Working With Source.....6

Handling New Upstream Versions.....7

Merging In Upstream.....7

Examining the Merge.....7

Committing the Merge.....7

Finalizing New Debian Versions.....8

Finding Historic Versions.....8

Fetching Source from History.....8

A. Command Reference.....9

hg-importdsc.....9

hg-importorig.....10

hg-markdeb.....12

hg-buildpackage.....14

Index.....17

Chapter 1. Introduction

Welcome to hg-buildpackage, a system that integrates the Debian[1] package build system with Mercurial[2].

Purpose

hg-buildpackage is designed to do this for you:

- Ability to track changes to both upstream and Debian sources through time
- Ability to retrieve any version of the sources from the archive, and build Debian binary and source packages from them.
- Generated packages can be built and used by people that have neither Mercurial nor hg-buildpackage installed.
- Provide an alternative to systems like dbs and dpatch to help manage patches that Debian applies to packages.
- Provide a way to ease collaboration between multiple Debian developers on a single package.
- Provide a way to use the merging and brancing features of Mercurial to help merge Debian changes into new versions of upstream packages.
- Make it easy to try hg-buildpackage and switch back to normal systems if you don't like it.

Features

hg-buildpackage has these features:

- hg-importdsc will import an entire Debian source package into your Mercurial repos, handling both upstream and Debian trees automatically. hg-importdsc can be run repeatedly to import newer versions, maintaining a full change history as it goes along.
- hg-importorig can import an upstream tar.gz or a directory into your archive with a single command.
- hg-buildpackage can use an existing orig.tar.gz, or can build you one from the Mercurial archive if necessary.
- hg-buildpackage works for both Debian-native and normal (upstream) packages.
- hg-buildpackage ensures that Mercurial metadata the (.hg and .hgtags) doesn't show up in tar.gz or diff.gz files.
- Tight integration with hg_load_dirs for intelligent handling and versioning of file renames upstream.
- Autosensing of information wherever possible from Debian changelogs, dsc files, and the Mercurial environment.
- hg-markdeb can be used to easily checkpoint development into Mercurial with tags.

[1] http://www.debian.org/
[2] http://www.selenic.com/mercurial

Usage Overview

Here is a high-level overview of the Debian development process using hg-buildpackage.

You'll start by creating your Mercurial repositories. You'll usually have to create two: one for the package, and a second for the upstream. By convention, these are named *`package`* and *`package`*.upstream.

Next, it's time to bring your packages into Mercurial. If you have some existing Debian packages, you'll import as much history as you like by using hg-importdsc. Otherwise, to start a new package in hg-buildpackage, you'll use hg-importorig to bring in the upstream sources. Then, you'll use **hg pull -u** to copy the upstream area to the Debian area.

As you work on your Debian sources, you'll use hg-buildpackage in place of debuild or dpkg-buildpackage (hg-buildpackage passes all its arguments to debuild for you). You can use standard Mercurial commands, such as **hg commit** or **hg log**, to commit changes and view history. When you have a final version of a Debian package, you'll upload it and run hg-markdeb to tag the version for future reference.

Chapter 2. Getting Started

Getting started with hg-buildpackage is pretty simple. This chapter shows you how.

The hg-buildpackage Repository Layout

hg-buildpackage assumes that you have two "canonical" repositories for every Debian package. These repositories correspond to the upstream and the Debian versions of the package's source. The upstream repository tracks unmodified upstream versions. The Debian repository pulls in upstream patches, and also has the Debian-specific patches for your package. (Debian-native packages have no upstream version and thus no upstream repository.)

Upstream Repository

The upstream repository tracks changes to the upstream program. You can use hg-importorig or hg-importdsc to import upstream sources from tarballs, or you can use Mercurial to track a Mercurial-based upstream with **hg clone** and **hg pull**.

Each upstream version should be tagged with a tag nammed UPSTREAM_*packagename_version*. Note that the Debian revision doesn't appear in this tag. An example would be UPSTREAM_mypackage_1.0.5. If you use hg-importorig or hg-importdsc, this tag will be created for you automatically. If you manage your upstream repository manually, you should create the tag yourself using **hg tag**

Debian Repository

The Debian repository tracks the Debian-specific package. The Debian package is represented as the upstream source plus a set of Debian patches. Upstream patches are normally pulled into the Debian repository before beginning on the Debian package.

Each Debian version carries a tag similar to the upstream version, but the Debian version tags begin with DEBIAN_ and contain a full Debian revision. An example is DEBIAN_mypackage_1.0.5-3.

Chapter 3. Importing Packages

You will need to import sources into your hg-buildpackage archive on several different occasions:

- You have existing Debian packages that you would like to maintain with hg-buildpackage. In this case, you'll want to import one or more full Debian source packages with **hg-importdsc**.
- You want to package some previously-unpackaged software for Debian. In this case, you'll want to import an upstream tarball with **hg-importorig**.
- You want to update your Debian package with a new version of upstream source. This case also calls for **hg-importorig**.
- You want to import Debian sources from someone else (for instance, if someone NMU's a package you maintain) into your tree. **hg-importdsc** can handle that.

There are two different programs that handle importing: **hg-importdsc** and **hg-importorig**. Both are covered in this chapter.

Importing Debian Source Packages

Importing a package is very easy. All you have to do is run **hg-importdsc** with the name of a .dsc file to import:

```
$ hg-importdsc ~/dpkg/rdiff-backup_0.12.3-1.dsc
FIXME: show output
```

In this example, I had never imported rdiff-backup before. **hg-importdsc** therefore initialized the repository for me. It then added and committed the upstream version (the first commit message), committed the Debian diffs (second commit), and noted which version in the archive corresponds to 0.12.3 and 0.12.3-1 (third commit message). (FIXME: these are probably old)

I can run it again with a new version:

```
$ hg-importdsc ~/dpkg/rdiff-backup_0.12.5-1.dsc
FIXME: show output
```

hg-importdsc is smart enough to know not to import an upstream version twice. But it will import the new Debian version.

hg-importdsc has a few restrictions: you must always load packages in ascending order of package version. Please see hg-importdsc(1) for more details.

Importing Upstream Sources

While **hg-importdsc** can solve many problems, sometimes you need to import just upstream sources (the Debian orig.tar.gz file). For this task, **hg-importorig** exists. You will often use it if you have already loaded all the Debian versions into your archive, upstream has released a new version, and you want to package that version.

Using **hg-importorig** is simple and straightforward; please refer to hg-importorig(1) for more details.

Chapter 4. Common Tasks

This chapter describes how to use hg-buildpackage to carry out regular, every-day development activities.

Building and Working With Source

Before you start working on any source package with hg-buildpackage, make sure you have your directories set up. hg-buildpackage assumes you will have two Mercurial branches for your package, each in its own repository, both stored in the same directory. (You can use symlinks if you wish to store them elsewhere).

If you package is named hg-buildpackage, your repositories should be named hg-buildpackage and hg-buildpackage.upstream. Just "hg init" both of them to get going.

Once you have sources, imported from another Mercurial tree, or imported with hg-importdsc or hg-importorig, you can start hacking. As you work, you will use **hg commit** to commit your changes to the Debian repository. (If you just want to build an old package, there’s no need to do this.)

When you’re ready to build a package, just run **hg-buildpackage** to build it. The **hg-buildpackage** command takes the same arguments as **debuild**, and passes them on to **debuild**, so give it your usual set. **hg-buildpackage** will use your existing orig.tar.gz file for source generation, if it exists; otherwise, it will take care of automatically building it from the upstream sources if necessary.

Here’s an example:

```
~/hg$ hg init bacula
~/hg$ hg init bacula.upstream
~/hg/bacula$ hg-importdsc ~/work/bacula_1.38.8-0.1.dsc
 *** Processing upstream file /home/jgoerzen/work/bacula_1.38.8.orig.tar.gz
adding ABOUT-NLS
adding COPYING
adding ChangeLog
...
/home/jgoerzen/hg/bacula/„hg-importorigtargz-aNFzUc9794/bacula-1.38.8 imported
into /home/jgoerzen/hg/bacula.upstream
 *** Upstream processing of /home/jgoerzen/work/bacula_1.38.8.orig.tar.gz done.
 *** Processing Debian source tree for /home/jgoerzen/work/bacula_1.38.8-0.1.dsc
 Pulling upstream into Debian tree
pulling from /home/jgoerzen/hg/bacula.upstream
requesting all changes
...
dpkg-source: extracting bacula in bacula-1.38.8
dpkg-source: unpacking bacula_1.38.8.orig.tar.gz
dpkg-source: applying /home/jgoerzen/work/bacula_1.38.8-0.1.diff.gz
adding debian/Makefile
adding debian/RATIONALE
adding debian/README.Debian
...
DSC successfully imported!
... now import some more versions ...
~/hg/bacula$ hg-buildpackage -rfakeroot
```

In this case, I had not checked out the upstream source and did not have my orig.tar.gz file handy. Therefore, **hg-buildpackage** checked out the upstream sources for me, generated the tar.gz file, and then called **debuild** to do the rest.

If you are building for Debian, you will want to keep your orig.tar.gz around so that future source uploads use the same MD5 sum in the .dsc file.

For more details, please see the manpage for hg-buildpackage(1).

Handling New Upstream Versions

A common scenario for a Debian developer to deal with is that of a new upstream release. The Debian patches from the most recent Debian release must be merged into the upstream one.

There have been a few tools to do that: **uupdate** is one. However, now that you are using Mercurial, you can use its built-in **pull** command to make this easier.

The first thing that you will do is to import the new upstream sources into your hg-buildpackage archive. Please see the Section called *Importing Upstream Sources* in Chapter 3 for instructions.

Next, check out the latest Debian version if you don't already have it. cd into your Debian repo.

Merging In Upstream

Now, you are ready to merge in the new upstream. Run **hg fetch**, giving it the path to your canonical upstream repository.

FIXME: add a sect3 demonstrating

Examining the Merge

Now, you will want to examine the merge, especially if Mercurial complained of any conflicts.

You'll also want to note the new version in `debian/changelog` and perhaps rename your directory based on the new version. I find it easy to run a command like **debchange -v 0.13.3-1**, then modify the changelog as appropriate. debchange will handle the rename for you.

Committing the Merge

Finally, you will want to commit the merge. If you want to just use a simple log message, a command like this will work:

```
$ hg commit -m "Merged in upstream 0.13.3"
FIXME: show output
```

The string supplied after the -m is the log message.

If you want to add a more detailed log, try this, just run **hg commit** and it will prompt you for one.

You can see what happened by using **hg log -vp**:

```
$ hg log -vp
FIXME: show output
```

Slick -- it shows exactly which upstream patches you used.

Finalizing New Debian Versions

When you have uploaded a Debian package to the archive, you should ask Mercurial to note this for you. That way, you can request this specific version later. Just run this:

```
$ hg-markdeb
FIXME: show output.
```

The reason for this is that you might make several commits during the course of hacking on a given Debian version. This command lets you note the final version, and to run it again, you must update the changelog.

Technically speaking, this creates the DEBIAN_ tag.

Finding Historic Versions

To find the versions available in your Mercurial archive, first cd to its top level and then run:

```
$ hg tags
FIXME: show output
```

Fetching Source from History

Use a command such as hg clone -rDEBIAN_package_1.3.5

Appendix A. Command Reference

hg-importdsc

<jgoerzen@complete.org>

John Goerzen

Name

`hg-importdsc` — Import a Debian source package into hg-buildpackage archive

Synopsis

hg-importdsc *`dsc_name`*

Description

hg-importdsc imports a Debian source package into an hg-buildpackage archive, notes the package version in the configs, and commits the change. All information, including package name, version, Debian diffs, and upstream source, is automatically detected from the source package.

This program will automatically detect if the upstream sources have already been imported, and if so, will not attempt to re-import them again.

It is an error to run **hg-importdsc** on a package whose Debian version already exists in the archive. It is also an error to run **hg-importdsc** on a package when a newer version already exists in the archive. **hg-importdsc** detects both conditions and will terminate without causing harm.

Options

dsc_name

Gives the path to the Debian .dsc file from the source package.

Bugs

- Neither GPG signatures nor MD5 sums are verified when unpacking Debian source packages.

Environment

This application uses no environment variables directly, but other programs it calls may do so.

Copyright

Author

hg-buildpackage, its modules, documentation, executables, and all included files, except where noted, was written by John Goerzen <`jgoerzen@complete.org`> and copyright is held as stated in the COPYRIGHT section.

hg-buildpackage may be downloaded from the Debian packaging system or at http://packages.debian.org/hg-buildpackage.

hg-buildpackage may also be downloaded using Mercurial by running **hg clone http://hg.complete.org/hg-buildpackage**.

See Also

Detailed usage information, including a background and a description of how the tools fit together, can be found in the hg-buildpackage Manual. On Debian systems, you may find this at `/usr/share/doc/hg-buildpackage`.

hg-buildpackage applications have references in that manual and the following manpages: hg-importdsc(1), hg-importorig(1), hg-markdeb(1), hg-buildpackage(1).

Other related references include: hg(1), dpkg-source(1).

hg-importorig

<jgoerzen@complete.org>

John Goerzen

Name

hg-importorig — Import an upstream source into a hg-buildpackage archive

Synopsis

hg-importorig *filename_or_dir package version*

Description

hg-importorig imports the tar.gz, tar.bz2, or zip file given by filename (or a directory, if filename is one) into the hg-buildpackage archive. Since upstream tarballs do not contain the package and version information in a unified format, you must specify that on the command line as well.

It is an error to run **hg-importorig** on a package when a newer version already exists in the archive. **hg-importorig** detects this condition and will terminate without causing harm.

Options

filename_or_dir

Gives the path to a tar.gz file to import.

Alternatively, gives the path to a directory to import without unpacking anything.

package

Gives the Debian source package name corresponding to this upstream package.

version

Gives the package version corresponding to this upstream package. This should not contain the Debian part (-1, -2, etc).

Environment

This application uses no environment variables directly, but other programs it calls may do so.

Copyright

Author

hg-buildpackage, its modules, documentation, executables, and all included files, except where noted, was written by John Goerzen <`jgoerzen@complete.org`> and copyright is held as stated in the COPYRIGHT section.

hg-buildpackage may be downloaded from the Debian packaging system or at http://packages.debian.org/hg-buildpackage.

hg-buildpackage may also be downloaded using Mercurial by running **hg clone http://hg.complete.org/hg-buildpackage**.

See Also

Detailed usage information, including a background and a description of how the tools fit together, can be found in the hg-buildpackage Manual. On Debian systems, you may find this at `/usr/share/doc/hg-buildpackage`.

hg-buildpackage applications have references in that manual and the following manpages: hg-importdsc(1), hg-importorig(1), hg-markdeb(1), hg-buildpackage(1).

Other related references include: hg(1), dpkg-source(1), tar(1).

hg-markdeb

<`jgoerzen@complete.org`>

John Goerzen

Name

`hg-markdeb` — Mark the Debian version of the working directory in archive

Synopsis

hg-markdeb

Description

hg-markdeb is used to tell hg-buildpackage that the version of a package represented in the current working directory is "official". The progam will find the Debian version from `debian/changelog` and record it as appropriate. No input is required and no arguments are required. **hg-markdeb** will abort if it is run more than once for a single Debian version.

Environment

This application uses no environment variables directly, but other programs it calls may do so.

Copyright

Author

hg-buildpackage, its modules, documentation, executables, and all included files, except where noted, was written by John Goerzen <`jgoerzen@complete.org`> and copyright is held as stated in the COPYRIGHT section.

hg-buildpackage may be downloaded from the Debian packaging system or at http://packages.debian.org/hg-buildpackage.

hg-buildpackage may also be downloaded using Mercurial by running **hg clone http://hg.complete.org/hg-buildpackage**.

See Also

Detailed usage information, including a background and a description of how the tools fit together, can be found in the hg-buildpackage Manual. On Debian systems, you may find this at `/usr/share/doc/hg-buildpackage`.

hg-buildpackage applications have references in that manual and the following manpages: hg-importdsc(1), hg-importorig(1), hg-markdeb(1), hg-buildpackage(1).

hg-buildpackage

<jgoerzen@complete.org>

John Goerzen

Name

`hg-buildpackage` — Build Debian packages from a Mercurial archive

Synopsis

hg-buildpackage [*`args for debuild`*...]

Description

The **hg-buildpackage** command is used to build Debian source and .deb packages from a checked-out version of a Debian source tree. Please refer to the hg-buildpackage manual for instructions on how to check out such a tree.

hg-buildpackage will, in order:

- Verify that it is being executed from the proper location.
- Check to see if a proper orig.tar.gz exists. If not, it will check out the proper upstream directory from the hg-buildpackage repository (if necessary) and build the orig.tar.gz. The orig.tar.gz will not contain Mercurial meta-data.

  If the upstream repository is not available locally, it will consult mirrors in the same manner as hg-get(1).
- Call debuild(1) (or the application specified in DBP_BUILDER) with arguments instructing it to ignore Mercurial meta-data in the diff.gz, passing along all arguments given to **hg-buildpackage**.

Options

All options passed to **hg-buildpackage** are sent directly to debuild(1).

ENVIRONMENT VARIABLES

If the DBP_BUILDER environment variable is set, it will be taken as the name of a command to use to build the Debian packages. It will be passed the same arguments as debuild would have received.

Example

```
~$ cd ~/tree/debian
FIXME: show example
```

Environment

This application uses no environment variables directly, but other programs it calls may do so.

Copyright

Author

hg-buildpackage, its modules, documentation, executables, and all included files, except where noted, was written by John Goerzen <`jgoerzen@complete.org`> and copyright is held as stated in the COPYRIGHT section.

hg-buildpackage may be downloaded from the Debian packaging system or at http://packages.debian.org/hg-buildpackage.

hg-buildpackage may also be downloaded using Mercurial by running **hg clone http://hg.complete.org/hg-buildpackage**.

See Also

Detailed usage information, including a background and a description of how the tools fit together, can be found in the hg-buildpackage Manual. On Debian systems, you may find this at `/usr/share/doc/hg-buildpackage`.

hg-buildpackage applications have references in that manual and the following manpages: hg-importdsc(1), hg-importorig(1), hg-markdeb(1), hg-buildpackage(1).

Other related references include: hg(1), debuild(1), dpkg-buildpackage(1).

Index

C

Configs, 8, 8

H

hg-buildpackage, 6
  Reference For, 14
hg-importdsc, 4
  Reference For, 9
hg-importorig, 4
  Reference For, 11
hg-markdeb, 8
  Reference For, 12

O

Overview, 2