Assuring the Quality of your Debian Packages

Ana Guerrero Lopez

March 16, 2004
MiniDebConf Barcelona 2014

Outline

What's this talk about?

- The Debian Quality Assurance (QA) Team has different systems for automatically checking the packages included in the Debian archive.
- On the other hand, there are QA checks you can do to your packages before uploading them to the archive. This talk is about these.

The goal and motivation of this talk

My goal is to give people some suggestions about how checking their packages beyond just running lintian. Hopefully everybody will learn at least a new trick during the talk!

I have found often myself explaining parts of this to people who were starting to make their first Debian packages.

Stages of checking

- Source packages (`*.orig.tar.gz/bz2/xz *.dsc *.debian.tar.gz` files)
- Binary packages. (`*.deb` files)
- Integration of your packages in the archive

pbuilder/cowbuilder

- pbuilder can create and maintain Debian chroot enviroments.
- It's an easier way to build in a clean and minimal environment
- It also checks that all the build dependencies are listed
- cowbuilder is a wrapper around cowbuilder using copy-on-write, which makes it faster.

pbuilder/cowbuilder

- Create a cowbuilder instance ("sid" by default):

```
# cowbuilder --create --mirror http://http.debian.net/debian
```

- Before every build, update your cowbuilder instance:

```
# cowbuilder --update
```

- Build your package

```
# cowbuilder --build yourpackage_2.3-1.dsc
```

pbuilder/cowbuilder

- If you don’t know how to create the dsc file, use:

```
$ dpkg-source -b yourpackage-2.3-1
```

- Or you also can use:

```
$ debuild -S
```

- More information `man pbuilder` and `man cowbuilder`

debdiff between source packages

- If you're updating a package it's very useful comparing the updated version with the previous one to check easily the changes you've made.

```
$ debdiff fabric_1.8.2-1.dsc fabric_1.8.2-2.dsc
diff -Nru fabric-1.8.2/debian/control fabric-1.8.2/debian/control
--- fabric-1.8.2/debian/control 2014-02-22 23:53:44.000000000 +0100
+++ fabric-1.8.2/debian/control 2014-03-14 15:16:41.000000000 +0100
@@ -11,7 +11,7 @@

 Package: fabric
 Architecture: all
-Depends: ${misc:Depends}, ${python:Depends}, python-paramiko (>= 1.6), python-pk
+Depends: ${misc:Depends}, ${python:Depends}, python-paramiko (>= 1.10), python-p
 Suggests: libjs-jquery
 Description: Simple Pythonic remote deployment tool
  Fabric is designed to upload files and run shell commands on a number of
```

lintian

- Not going to explain this one :)
- Recommend using `lintian -Ii *.changes`
  - `-i` Print explanatory information about each problem discovered
  - `-I` Display informational `I:` tags as well.

debdiff between binary packages

- If you're updating a package it's very useful comparing the updated version with the previous one to check easily the changes you've made. Note that we're comparing now binary packages.

```
$ debdiff fabric_1.8.2-1_all.deb fabric_1.8.2-2_all.deb
File lists identical (after any substitutions)

Control files: lines which differ (wdiff format)
------------------------------------------------
Depends: python (>= 2.7), python (<< 2.8), python-paramiko (>= [-1.6),-] {+1.10),
Version: [-1.8.2-1-] {+1.8.2-2+}
```

dpkg

There are 3 very simple checks you should do with dpkg:

- `dpkg -c fabric_1.8.2-2_all.deb` Check that the package doesn't install spurious files and nothing important is missing.
- `dpkg -I fabric_1.8.2-2_all.deb` Check if the Depends line is OK. And your conffiles or maintainer scripts were added.
- `dpkg -i fabric_1.8.2-2_all.deb` Yeah, install it!
  - It might sound stupid, but some people test the software before packaging and they don't test the final install.
  - If you can test it in a virtual machine (e.g. VirtualBox) with the minimum software installed, even better. It'll help you to find missing running dependencies.

adequate

- Young tool but it's promising.
- It's similar to lintian but it runs on installed packages
- You can run it once your package is installed:

```
$ adequate gimp
gimp: py-file-not-bytecompiled /usr/lib/gimp/2.0/plug-ins/pyconsole.py
```

- More information `man adequate`.

piuparts

- "package installation, upgrading, and removal testing suite"
- It's a tool for testing that .deb packages can be installed, upgraded, and removed without problems.
- It's as easy as:

```
piuparts kdbg_8.9-1_amd64.deb
```

- However, this will take some time because it creates a sid chroot with debootstrap, where it'll test your package. If you want to use a premade tarball or test with another suite, there is a handy 5-minutes tutorial at: `https://piuparts.debian.org/doc/README_1st.html`

Checking with your local repository

- You can create your own local Debian repository to check the package updagres and removal.
- This is compulsory when your package has maintainer scripts or when your package has multiple binaries.
- You can use apt-ftparchive or reprepro.