Manage your Invenio local patches using guilt

my share of idioms, tricks and lessons learned

Invenio User Group Workshop 2013

Forschungszentrum Jülich

18-20 November 2013

Ferran Jorba

Universitat Autònoma de Barcelona

Ferran.Jorba@uab.cat

The problem to solve

```
$ make install
[...]
 /bin/mkdir -p '/home/ddd/invenio/etc/build'
 /usr/bin/install -c config.nice '/home/ddd/invenio/etc/build'
for d in / /cache /log /tmp /data /run ; do    \
        mkdir -p /home/ddd/invenio/var$d ;        \
    done
**************************************************************
** Invenio software has been successfully installed!        **
**                                                          **
** You may proceed to customizing your installation now.    **
**************************************************************
```

Repeat that:

- for each installation (test and production).
- for each new release.

Free software, local patches

- The freedom to study how the program works, and change it so it does your computing as you wish (freedom 1). (http://www.gnu.org/philosophy/free-sw.html)
- Your local changes are patches. They are born small, but they grow, and grow.
- Either you educate them since the beginning, or when they become teenagers, you are lost.
- Invenio is free software, so better begin now.

Needs and wishlist

- Need to save our customizations, and replicate them after each new Invenio release.
- Customizations grow with your site, even if you are eager to integrate them upstream.

- Easy to understand and use.
- Easy to undo errors.
- Easy to replicate our customizations in all our instances; industrial-style solution

The birth of our tool: a stack of patches

2002: Andew Morton patch management scripts (http://lwn.net/Articles/13518/)

- The key philosophical concept is that your primary output is patches. [...] So patches are the first-class object here.
- These scripts are designed for managing a "stack" of patches. [...] They're pretty fast, and simple to use.

A bright idea with many reimplementations

- **2003: quilt, by Andreas Grünbacher.**
- 2005: git, by Linus Torvalds.

Quilt on top of git:

- 2005: stgit, by Catalin Marinas.
- **2006: guilt, by Josef Sipek.**
- 2008: topgit, by Petr Baudis.
- [...]
- 2013: git-queue, by Morita Kazutaka.

Similar tools for Mercurial.

Why not just quilt?

- quilt needs to know beforehand which files you are going to modify, otherwise it just fails. Not obvious if you use Invenio web front-end or are a newcomer.
- guilt knows it automatically because git automatically detects changed files. You only need to inform about new files (and git helps you here too with `git status`).
- All git commands, helpers and tools are available, if needed (grep, tig, gitk, etc.)

Why not just git?

- Not so easy to undo and redo.
- It covers other needs: software development, branches, integration…
- Maybe with branches...?
- I feel that it is easier and clearer to use a different tool designed just for that: *my patches over a third-party software*.

Why guilt, and not stgit, topgit, ...?

- I was used to quilt, and guilt keeps the same commands and flags (so I could just change my `alias q=quilt` to `alias q=guilt`).
- quilt is already quite a standard (ex. Debian source format 3.0 (quilt)), so there are tutorials and a community around.
- It is mature and (actively) maintained.
- After trying the others, I just didn’t like any so much.
- It works!

So, how does guilt handle patches?

- It keeps a linear history of patches.
- A patch is a set of modifications to files.
- You mostly work on your current (top of the stack) functionality (patch).
- If you mess it up, you undo (pop) it easily.
- When you need to work on something else, you create a new patch that pushes the older ones down to the stack (history).
- You can always undo (pop) one or all patches to go back to your pristine install.

Overview of our magic plan

1. Undo our customizations (patches).
2. Install new Invenio release.
3. Redo our customizations (patches).

```
install-dir$ guilt pop name-of-your-last-make-install-x.y.a.patch
install-dir$ guilt push # so it gets applied again
install-dir$ guilt new make-install-x.y.b.patch
build-dir$ make install
install-dir$ guilt refresh
install-dir$ guilt push --all
```

guilt: first time init

```
$ cd /your/invenio/install/dir
$ git init # First, init git
$ echo "*~
*.pyc
*.OLD
*.tmp
*.xml
var/" >.gitignore
$ git add --all
$ git commit
$ guilt init # And then, guilt
```

A common session (I): ~~customize~~ patch languages box

```
ddd@test:~/invenio$ guilt new put-languages-at-top.patch # Give a name to your patch
ddd@test:~/invenio$ git grep languagebox # Use git commands to help you
lib/python/invenio/webstyle_templates.py:           %(languagebox)s
lib/python/invenio/webstyle_templates.py:           'languagebox' : self.tmpl_language_selection_box(req, ln),
ddd@test:~/invenio$ emacs lib/python/invenio/webstyle_templates.py # Hack, test, hack, test...
ddd@test:~/invenio$ guilt files # see which files are modified by this patch
lib/python/invenio/webstyle_templates.py
ddd@test:~/invenio$ guilt diff # see what you have modified
ddd@test:~/invenio$ guilt refresh # commit
ddd@test:~/invenio$ guilt pop # undo
Now at use-img-uab-path-for-local-icons.patch.
ddd@test:~/invenio$ guilt push # redo
Applying patch..put-languages-at-top.patch
Patch applied.
```

A common session (II): create a new websubmit form

```
ddd@test:~/invenio$ guilt new create-movies-form.patch # Give a name to your patch
# work with Invenio WebSubmit web interface; clone forms,
# edit bibconvert rules, test, etc.
ddd@test:~/invenio$ git status # Use git commands to find which files are new or modified
# On branch invenio-1.1.0
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
[...]
ddd@test:~/invenio$ guilt add files # add newly created files
ddd@test:~/invenio$ guilt new something-else.patch # add something else
Uncommited changes detected. Refresh first.
ddd@test:~/invenio$ guilt refresh # commit
Patch create-movies-form.patch refreshed
ddd@test:~/invenio$ guilt new something-else.patch
```

A common session (III): upstream says: try this patch...

```
$ cd /tmp/patches
$ wget -O fix-my-bug.patch
http://invenio-software.org/repo/invenio/commit/?id=342b2e667d4…
# For each modified file, fix its path so it can be applied to your
installed dir...
$ sed -i s/dist-path.ext/install-path.ext/ fix-my-bug.patch
$ cd /your/invenio/install/dir
$ guilt import /tmp/patches/fix-my-bug.patch
$ guilt push # Try to apply it
$ emacs $(guilt next -p) # Maybe fix some path or else
$ guilt push # Try to apply it again
```

- If it fixed the problem, it is integrated into your history.
- If it didn't, let's delete it.

```
$ guilt pop # Undo
$ guit delete -f $(guilt next) # Delete it completely from history
```

A common session (IV): Send and apply the top patch to another host

- On test:

```
ddd@test:~/invenio$ guilt top -p
/home/ddd/invenio/.git/patches/invenio-1.1.0/improve-author-links.patch
ddd@test:~/invenio$ scp $(guilt top -p) ddd@prod:/tmp/patches
ddd@prod's password:
improve-author-links.patch                    100% 7976     7.8KB/s   00:00
```

- On production:

```
ddd@prod:~/invenio$ guilt import /tmp/patches/improve-author-links.patch
ddd@prod:~/invenio$ guilt push
Applying patch..improve-author-links.patch
Patch applied.
```

Troubleshooting (I): file does not exist and --remove not passed

```
traces@test:~/invenio$ guilt new remove-obsolete-authorities.patch
[Now use web interface to remove two files]
traces@test:~/invenio$ guilt files
etc/bibformat/output_formats/AB.bfo
etc/bibformat/output_formats/AD.bfo
traces@test:~/invenio$ guilt refr --diffstat
error: etc/bibformat/output_formats/AB.bfo:
       does not exist and --remove not passed
fatal: Unable to process path etc/bibformat/output_formats/AB.bfo
error: etc/bibformat/output_formats/AD.bfo:
       does not exist and --remove not passed
fatal: Unable to process path etc/bibformat/output_formats/AD.bfo
Patch remove-obsolete-authorities.patch refreshed
traces@test:~/invenio$ guilt top
remove-obsolete-authorities.patch
traces@test:~/invenio$ guilt pop
Uncommited changes detected. Refresh first.
```

Troubleshooting (I): guilt repair --full

```
traces@test:~/invenio$ guilt repair --full
Checking status file format...ok; no upgrade necessary.
Current  HEAD commit be7f2d1bed2d9804ad45b34f61843314ab430d09
New      HEAD commit 8c4a32aa80e5ec5b86917e53b637727f29a43ed1
About to forcefully pop all patches...
Are you sure you want to proceed? [y/N] y
Patches should be popped.
Repair complete.
traces@test:~/invenio$ guilt top
traces@test:~/invenio$ guilt push -a
[...]
Applying patch..urlutils.patch
Patch applied.
Applying patch..webinterface_handler_wsgi.patch
Patch applied.
Applying patch..websearch_templates.patch
Patch applied.
[...]
```

Troubleshooting (II): patch does not match the current contents

```
traces@test:~/invenio$ guilt push -a
[...]
Applying patch..bibindex_engine.patch
Patch applied.
Applying patch..webinterface_handler_wsgi.patch
Patch applied.
Applying patch..bfe_areatematica.patch
Patch applied.
Applying patch..remove-obsolete-authorities.patch
error: the patch applies to 'lib/python/invenio/ __init__.pyc'
 (5eddbfa3a8bd0a8799ced228ab6e6dfbba259c10),
  which does not match the current contents.
error: lib/python/invenio/ __init__.pyc: patch does not apply
 To force apply this patch, use 'guilt push -f'
```

Troubleshooting (II): edit the next patch to remove an unwanted hunk

```
traces@test:~/invenio$ guilt pop
Now at improve-author-links.patch
traces@test:~/invenio$ guilt next -p
/home/traces/invenio/.git/patches/invenio-1.1.0/remove-obsolete-authoritie
s.patch
traces@test:~/invenio$ emacs $(guilt next -p)
diff --git a/lib/python/invenio/__init__.pyc
b/lib/python/invenio/__init__.pyc
index
ee8a6360c37b9326c93e172c292def235a939b50..5eddbfa3a8bd0a8799ced228ab6e6dfb
ba259c10 100644
GIT binary patch
delta 16
Xcmcb@c7=`o;wN4%jmvQx*?pM IBo^^

delta 16
Xcmcb@c7=`o;wN6NQ?BY8*?pM ICTZo
[remove those offending lines]
```

Troubleshooting (III): patch does not apply

```
ddd@prod:~/invenio$ guilt import
/tmp/patches/use-img-uab-path-for-local-icons.patch
ddd@prod:~/invenio$ guilt push
Applying patch..use-img-uab-path-for-local-icons.patch
error: patch failed:
lib/python/invenio/bibformat_elements/bfe_fulltext.py:124
error: lib/python/invenio/bibformat_elements/bfe_fulltext.py:  patch does
not apply
To force apply this patch, use 'guilt push -f'
ddd@prod:~/invenio$ emacs $(guilt next -p) # edit at will
```

Lessons learned

- Pathes are done over intalled dir.
  - Otherwise, you don't integrate websubmit & etc.
- Choose a clear name for the first patch after an Invenio upgrade, like `make-install-1.1.1.patch`.
  - This will be the base to apply your patches next upgrade.
- Sometimes it is necessary to flatten patches.
  - After several months of modifying the same files here and there, history gets messy.
  - Chances are that after upgrading, patches will fail in a way difficult to decide the fix.

How to convert your customizations to patches or flatten history

Ok, say you are convinced. How to start with?

https://github.com/fjorba/localpatches

- It compares your built Invenio with your current installation, and creates a bunch of patches, one per file, so they can be saved and imported to guilt and applied after your next upgrade.
- It also flattens your guilt(y) history.

References, tutorials, etc

- How To Survive With Many Patches, or Introduction to Quilt: http://www.suse.de/~agruen/quilt.pdf
- Quilt for Debian Maintainers: http://pkg-perl.alioth.debian.org/howto/quilt.html
- Quilt tutorial: http://www.shakthimaan.com/downloads/glv/quilt-tutorial/quilt-doc.pdf
- A Guilty Git: http://kernelpanic.blogspot.com/2007/03/guilty-git.html

Thank you for your attention

Questions time!