

Merging branches and solving conflicts

Merge check-list:

1. Commit.
2. bzd merge ...
3. Solve conflicts (if any).
4. Write merge changelog.
5. Commit.

IMPORTANT

ALWAYS commit
just before *and* after merging!

Conflict check-list:

1. File myfile has a conflict
⇒ myfile.BASE: before divergence
⇒ myfile.THIS: local version
⇒ myfile.OTHER: from other tree
2. kdiff3 myfile.BASE myfile.THIS
↔ myfile.OTHER # One line only
3. Save result as myfile.
4. bzd resolve myfile
→ removes files created at step 1!

IMPORTANT

If *myfile* does not exist when typing
bzd resolved, it will automatically
be marked as *removed* by bzd.

Useful addresses

ABINIT Forge

bzd+ssh://archives.abinit.org/
↔ abinit/<version>/<developer>/<branch>/

*Development branches of ABINIT 5.5 and later. All of the
URL should be typed at once (one line only, no space).*

ABINIT Website - Developer's Corner

<http://abinit.org/developers/>
Reference information for developers.

Bazaar Website

<http://bazaar-vcs.org/>
Home page of the Bazaar Version Control System.

Mailing lists

Announcements

announce@abinit.org
*Low-traffic list for important announcements about
ABINIT.*

User Forum List

forum@abinit.org
Discussions about the use of ABINIT.

Developer Forum List

developer@abinit.org
Discussions about the development of ABINIT.

Committer List

gnuarch@abinit.org
*For developers having an access to the ABINIT Forge
(restricted).*

Need help?

To get help on all Bazaar commands, just remember
bzd help.

ABINIT FORGE



Quick reference for committers

Read this first

On <http://abinit.org/developers/>, in the *Bazaar* section:

- Introduction, part I
- Introduction, part II
- Introduction, part III
- Non-standard install

and documents cited therein.

Structure of the ABINIT Forge

Full URL of a branch (one line, no space):

```
bzr+ssh://archives.abinit.org/  
↳ abinit/<repository>/x.y.z-private
```

x: major version number
y: minor version number

repository: *trunk*, or committer login

Public branches (created automatically):

- x.y.0-public
- x.y.1-public
- x.y.2-public
- x.y.3-public
- x.y.4-public
- x.y.5-public

Private branches: automatically created replacing "public" by "private" in the names above; additional branches free-form.

Useful commands

bzr ...	Result
help	Get help
info	Get source tree info
status	Get status report

BEFORE ACCESSING THE FORGE

```
export EDITOR="/path/to/my/editor" (BASH)  
or  
setenv EDITOR "/path/to/my/editor" (CSH)
```

Working with branches

A *branch* is an autonomous copy of the source code. All history is kept locally, except when explicitly *published* by the committer.

Action	Command
Get	bzr branch url [local.dir]
Sync	bzr pull [url]
Publish	bzr push [url]
Off-line	bzr commit

Typical use: decentralised development.

If the working tree is empty, run `bzr checkout` from within.

Working with checkouts

A *checkout* is a branch the history is kept on the Forge. It can however be made autonomous temporarily.

Action	Command
Get	bzr checkout url [local.dir]
Sync	bzr update
Publish	bzr commit
Off-line	bzr commit --local

Typical use: one developer working on two computers.

Writing changelogs

Template:

One short summary line (no trailing dot)

* dir1/sub1/file1: Some changes. Make full sentences.

* dir2/sub2/file2,dir3/sub3/file3: Some other changes.

* dir4/sub4/file4: Related changes (no blank line before).

* Additional notes and issues.

GNU changelog format: please make sure that all lines are < 80 characters and start at the first column. For a complete reference (one-line URL, no space):

```
http://www.gnu.org/prep/standards/  
↳ html\_node/Change-Logs.html
```

Committing

Check-list:

1. `bzr status`
2. Process files marked as unknown. Go back to 1 until no file is marked as unknown.
3. Write changelog.
4. `bzr commit --strict [-F logfile]`
Use `-F` option if your changelog is in a file.

NOTE

Committing to a public branch will soon trigger a nightly build followed by a run of the test suite, informing the gatekeeper afterwards.