Bazaar tutorial for CCP4

Ville Uski

September 17, 2013

Contents

1 Introduction

This is a brief guide to using Bazaar (bzr) as a version control system with the CCP4 software suite in a UNIX-like environment[1]. Any feedback can be emailed to `ville.uski@stfc.ac.uk`. You will need a user account on the CCP4 server fg.oisin.rc-harwell.ac.uk.

There is also a quick reference available on the Bazaar website [1]. Full documentation can be found in [2] and details on installing bzr on various platforms in [3]. Help pages for individual bzr commands can be accessed on the command line, e.g.,

```
$ bzr help checkout
```

Each CCP4 module on the bzr server fg.oisin.rc-harwell.ac.uk has a directory, with a structure like this:

```
$ ls -A pointless
  .bzr pointless-1_6_4-local pointless-1_5_22-local trunk
```

where the directory `.bzr` contains a **shared repository**, which is shared by all **branches** of the module `pointless`. The shared repository has a database

[1] info for Windows to be added later on

with full revision history for all branches, which are included in the other subdirectories shown in the listing. The branch **trunk** always includes the latest development version of the module. Normally, developers would push their changes to the trunk, but they can also maintain their own branches locally before merging them with the trunk (see section 3).

In Bazaar, there are two different ways to use the repositories: checkouts and branches. These are briefly discussed in sections 2 and 3. But first, you should tell Bazaar your name:

```
$ bzr whoami "Your Name <your.name@example.ac.uk>"
```

The other commands won't work without this. This needs to be done only once, as the information is stored in the configuration file `~/.bazaar/bazaar.conf`.

Let us also define a variable BZRROOT:

```
$ BZRROOT=developername@fg.oisin.rc-harwell.ac.uk/var/lib/gforge/\
chroot/scmrepos/bzr
```

where 'developername' is your user account on the server. There is no need to define this variable in practice, but it's used in this document to shorten the command lines.

2 Checkouts

Bazaar checkouts may be most useful when the developer(s) want(s) to work directly on the trunk. They would create a local copy of the trunk, so that this copy is *bound* to the original branch on the server. This is similar to using CVS or Subversion, with an additional option of making local revisions before committing them to the server.

To check out the latest revision of the module pointless, do

```
$ bzr checkout bzr+ssh://$BZRROOT/pointless/trunk pointless
```

which creates a local directory pointless including the full revision history of the branch trunk (a so-called heavyweight checkout).

After giving the command, you are prompted for your password. You can also copy your public ssh key to the server, so that you don't need to type the password. Here is the URI for the page where the public keys can be uploaded:

```
https://fg.oisin.rc-harwell.ac.uk/account/editsshkeys.php
```

After working on the files in the directory, the changes are committed:

```
$ bzr commit
```

This will open the default editor for the user to add a description of the new revision. The changes are then pushed to both the local and remote repositories. The command should be run in the directory `pointless` (or its subdirectories).

If you made a mistake, you can **uncommit** the changes, or **revert** from any changes in the working tree to a previous revision.

To add a new file to the repository, use

```
$ bzr add [FILE...]
```

If 'FILE...' is omitted, all new files (including subdirectories and the files therein) are added. It's still necessary to commit the changes.

There are also commands to rename (`move`) or delete (`remove`) files or directories from the repository. Other frequently used commands include `log`, which displays the full revision history for the branch, and `info`, which tells the URI of the parent branch to which the local branch is bound to (if any). The parent branch can be changed by using the commands `unbind` and `bind`. **status** is another frequently used command. Use the help for details.

2.1 Anonymous checkouts

Above, I checked out pointless using a URI beginning with `bzr+ssh://`. An alternative URI is

```
https://fg.oisin.rc-harwell.ac.uk/anonscm/bzr/pointless/trunk
```

which would create an *anonymous* checkout. This is a read-only checkout, i.e., you cannot commit changes from it to the server. You can commit them locally, if you first run the **unbind** command on the local branch.

You can convert an anonymous checkout to a normal one by changing the URI of the parent repository, as follows:

```
$ bzr unbind
$ bzr bind bzr+ssh://$BZRROOT/pointless/trunk/
```

2.2 Local commits and lightweight checkouts

Commits can be done only locally by using `'bzr commit --local'` or unbinding the local branch (`'bzr unbind'`). After a series of local commits, it's again possible to commit non-locally by omitting the `--local` option or (if required) rebinding the local branch to the one on the server (`'bzr bind'`). Local commits are useful if working offline or simply to reduce a chance of a bad commit.

It is also possible to have lightweight checkouts (`bzr checkout --lightweight`) which will copy the working tree but not the revision history. Lightweight checkouts depend on access to the branch for every operation, and so don't support local commits. Otherwise, they work like 'heavyweight' checkouts, but require less downloading initially.

2.3 Resolving conflicts

If the branch on the server has already been modified by another user, Bazaar will make sure that the local repository is up to date before making the commit. In this case, you can run the command

```
$ bzr update
```

which will 'merge' your local working tree with the latest revision in trunk. Bazaar will notify the number of 'conflicts' occurred in the merging and indicate the files with two conflicting versions. The command

```
$ bzr diff
```

will tell exactly where the differing lines in each file can be found. You can then edit these files to your liking. You will notice that they include labels '<<<<<<< TREE', '=======' and '>>>>>>> MERGE-SOURCE'. Your versions of the differing lines are between '<<<<<<< TREE' and '=======', whereas the merged lines are between '=======' and '>>>>>>> MERGE-SOURCE'.

After editing each file, run

```
$ bzr resolve [FILE...]
```

which will declare the conflict(s) resolved. After this, you still need to commit the changes.

There are visual merge tools, like meld, xxdiff, kdiff3, opendiff, which can be used together with Bazaar, if you install the Bazaar plugin `extmerge`. There is more information on plugins in [2]. See also section 4.

3 Branches

Bazaar is a distributed version control system, such as Git or Mercurial. Such systems are normally used such that each developer work on their local branches, which are independent of each other, and not bound to any central repository as is the case with checkouts. This means the developer has to push his changes to the server after committing them locally. If there are several developers working in the same project, a frequent task is to merge together changes from two diverged branches with common ancestry (i.e., earlier revision history).

To create a local branch of the trunk, do, e.g.,

```
$ bzr branch bzr+ssh://$BZRROOT/pointless/trunk pointless
```

which 'clones' an independent local branch in the directory pointless. You may also want to clone that branch locally, for example, to try experimental features, or to have one branch to work on and another to synchronize with the server.

After working on the files in the directory, you can commit the changes in the usual way,

```
$ bzr commit
```

Then, you can push the changes to the server:

```
$ bzr push --remember bzr+ssh://$BZRROOT/pointless/trunk
```

The option `--remember` makes bzr remember the URI so you don't need to type it next time you push (`bzr info` will reveal the remembered push branch).

The push command will only work if the target branch has not diverged from the source. In that case, you could merge first:

```
$ bzr merge bzr+ssh://$BZRROOT/pointless/trunk
```

and then resolve the conflicts as discussed in section 2.3. Alternatively, you can use the `--overwrite` option to replace the other branch completely, discarding its unmerged changes.

4 Plugins

Plugins add advanced features to Bazaar. These can be installed either system-wide or per user in the `~/.bazaar/plugins` directory. Most plugins are available in Launchpad, Bazaar's social coding website. For example, to install the plugin loggerhead (a web viewer to show graphically the version history and other information on branches), do as follows:

```
$ cd ~/.bazaar/plugins
$ bzr branch lp:loggerhead
```

If you happen to use Ubuntu Linux, this and many other plugins can also be installed from the Ubuntu repository:

```
$ sudo apt-get install loggerhead
```

References

[1] Bazaar Quick Reference Card, `http://doc.bazaar.canonical.com/beta/en/quick-reference/index.html`

[2] Bazaar 2.4 documentation, `http://doc.bazaar.canonical.com/en/`

[3] Download and Install Bazaar, `http://wiki.bazaar.canonical.com/Download`