UCLALES Git cheat sheet

Thijs Heus

September 14, 2009

The UCLA LES is managed with git as a content management system. The main distinction of Git over (e.g.) SVN, is that git is distributed. This means that everyone could carry all information at his local git repository. If you have no internet, you can work on without a problem, if somebody screws up the main server, any copy from any of the users would suffice to get everything correcct again. Theoretically, we do not even need to set up a main git server, but it is nice to have one and refer to it as 'the main code'. Therefore, our code is located at www.gitorious.org/uclales Everybody can download it, and remain up to date, but for developers it is necessary to set up a few more things.

1. Make sure you have git installed on your local system.
2. Set up an account at gitorious.org
3. Set up a local repository

1 How to set up an account?

First go to www.gitorious.org and set up an account over there. Then ask someone who is already part of uclales-dev to add you to that group. Now you could edit all the repositories. There is one master repository that represents the main code, and one branch per developer where everyone is free to do in what he wants. Of course, other peoples branches should normally not be touched, and the master branch with much care, that is, for bugfixes and clear features that are well developed only. Pushing code to gitorious goes best over ssh, and to get that working smoothly you can add your ssh key (located in `~/.ssh/id_rsa.pub`) to your gitorious profile.

Once you have completed your gitorious account, you can get the code by typing

```
git clone git://gitorious.org/uclales/uclales.git uclales
```

at the command line. This gives you the master branch. If you want to switch to a different branch that is already made, the first time you type:

```
git checkout -b newbranch original/newbranch
```

After that, changing branches can be done with:

```
git checkout branchname
```

2 Usefull commands

The man pages of git are not very clear sometimes, but they are quite exhaustive. Be sure to give them a look. Updating your local repository with remote changes

goes with git pull:
`git pull branchname origin`
to update just one branch, or just
`git pull`
to update everything. If you want to commit your changes to your (local) repository, type
`git commit -a`
and describe what you are changing in the commit file. It is advisable to only commit if a feature/bugfix is in principle complete, and especially to keep the code in the repository compilable at all times. Putting stuff in the remote repository goes with
`git pull`
If you want to check what has been changed since the last commit, use
`git status` and `git diff`
Adding files to, removing from and moving inside the repository needs to be done carefully: Use
`git add, rm` or `mv`
for that.

If you want to merge the changes of branch1 into branch2, do:
```
git checkout branch2
git merge branch1
```
If there are merge conflicts (could also happen with a
`git pull`),
then you can use
`git mergetool`
to resolve the conflicts manually.

If there is a specific commit you would like to have merged into your branch, you can look up the commit number (e.g. at gitorious) and use cherry-pick:
`git cherry-pick 6159a97`
If you screwed up the code and want to get back to your last commit:
`git reset --hard`

A faulty commit can be reset with
`git reset -soft HEAD^`
In general, be careful with reset and especially reset –hard, since it can confuse the history of the commits. To remove all files and directories that are not tracked by git from the current dir, use
`git clean -d -f`