git: version control and collaboration

A. J. Roberts

December 1, 2011

The software package git empowers you to easily maintain a history of a project, so you can recover old information for example, and additionally to collaborate with others on the project, so you can all work in parallel for example. These two aspects are introduced separately:[1] Section 2 introduces how to maintain a local history; and Section 3 introduces collaboration over the internet.

1 Installation and initialisation

Download Perhaps download git from `http://git-scm.com/download` To install on a Mac: download the `dmg` file; double click to unpack; look in the Git 'disk symbol' on the desktop; double click on the `pkg` file, and follow instructions. This installs git software accessible as commands from any terminal window.

Introduce yourself to git Set some useful global parameters for git. In a terminal window type the following commands[2]

[1] I base this introduction on *Git for the lazy*, `http://www.spheredev.org/wiki/Git_for_the_lazy` [Nov 2011].

[2] The first two establish your identity; the third and fourth invoke some pretty colouring for the command interaction; and the last changes the default editor for command interaction to something that is generally easier to use (nano is a free version of pico).

```
git config --global user.name="Your Name"
git config --global user.email="youremail@adelaide.edu.au"
git config --global color.ui=auto
git config --global color.interactive=auto
git config --global "nano -w"
```

Review these settings, at any time, by executing in a terminal window the command

```
git config --global --list
```

Perhaps use a graphical user interface Many people want a graphical user interface to git. The web has many suggestions, and differing opinions. I tentatively suggest using 'git gui', but use another GUI if you prefer. This application comes with git on common operating systems: just type `git gui &` into a terminal window.[3]

2 Manage your project locally

With git there are *three* places for storage of your files, there is:

- your working area, directory, where you work, edit and refine files in your project as usual;
- a local store of history of the project and its stages that is managed by git; and
- possibly an online repository, also managed by git, where you and collaborators merge independent progress on the project.

This second section only addresses the first two aspects of you working alone, with git, in your local working and storage area. Section 3 discusses collaboration over the internet.

[3] Make a desktop shortcut to git-gui by creating a file, named `gitgui.command`, with two lines: first, `#!/bin/bash`, second `git gui &`. Then execute in a terminal window, `chmod ugo+x gitgui.command`. Subsequent double-clicking the file icon should start git-gui.

2.1 Start your project

Make your working directory for the project as usual, say called `myproject`. Then execute the following commands in a terminal window.

`cd myproject` First change to the directory in which you are going to work.

`git init` Tell git to start managing the history of selected files in this directory.[4]

`git add .` Tell git to manage all the current files in the directory; or instead of the dot, specify a list of specific files.

`git commit` Stores the information that this (first) version is an identifiable stage, a 'milepost', in the project. git will request you type a message in the editor: make the first line a one line overall summary, such as "Initial version", and optionally provide additional information in subsequent lines.

Equivalent Git-gui

- Initialise the existing directory for git by selecting `Create new repository` from the the dialogue box, browse to the `myproject` directory, then click `Create`.
- To add files to be managed: click on the *icon* of the required file(s) in the top-left-pane (so they move to the bottom-left); whereas clicking on the file*name* displays their contents in the top-right window.
- To commit: enter a message such as "Initial version" in the window at the bottom=right; optionally click `Sign Off`; and finally click `Commit` button at the bottom.

[4] git creates an invisible sub-directory called `.git` in which is stored the history of the stages in your project. Do not meddle with this sub-directory.

2.2 Work in small stages

It is best to work in small stages: remember, if you cannot summarise the work in the last stage in a one line sentence, then you have gone too long without committing. Typically work according to the following cycle, repeat as much as you like but ensure you end with the `add` and `commit`.

Working Work, edit and refine files in your project as usual.

`git status` Optional, checks which files you have changed.

`git diff` Optional, check what the actual changes were.

`git add file1 file2` Essential, nominates the files (and perhaps new files and new directories) whose updates are to be saved as the new version at this stage.

`git commit` Essential, commits the current version of your nominated files as an identifiable stage in your project; enter and save your commit message (of at least a one line summary).

Equivalent Git-gui

- Work, edit and refine files in your project as usual, in small stages.
- Start git-gui and choose the project directory from the `Open Recent Repository` list in the opening dialogue box. If git-gui is already executing, then click `Rescan` to update git-gui's information about your repository.
- Managed files you have edited will be flagged by a blue icon; unmanaged files are also listed; managed files that have not changed are not listed.
- Optional, click on the file*name* in the top-left-pane to see, in the top window, the `diff`erences between the current version and the last commit.
- Click on the *icon* of changed files in the top-left to identify (add) them as the ones that will be committed (those appearing in the bottom-left).

Alternatively, 'add' all changed files under management by choosing `Stage Changed Files to Commit` from the `Commit` menu.

- To commit: enter a message in the middle window at the bottom; and finally click the `Commit` button at the bottom.

2.3 Review your work

`git log` To overview history so far of the project.

In git-gui, click on the `View master's history` from the `Repository` menu.[5]

`git log --pretty=oneline` Lists the one line summaries.

`git commit --amend` Changes the message of the last commit.

In git-gui, click on the `Amend Last Commit` box on the middle-right.

`git reset --hard` If you have not committed, but realised that you have messed up your local files, then use this to recover the files as at the last commit.

`git checkout filename` Just recovers the named file as at the last commit.

`git mv oldname newname` Version control requires a little discipline. One is that files under version control must only be renamed/moved via git using this command.

`git rm filename` Similarly delete/remove files under version control only via git using this command.

3 Collaborate over the internet

Yet to do.

[5] This view uses gitk. If gitk fails to start, you may need a soft link such as: `cd /usr/local/bin`, then `su ln -s /usr/local/git/bin/gitk`.