git: version control and collaboration

A. J. Roberts

December 5, 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: Section 2 introduces how to maintain a local history;[1] 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 core.editor="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 GitHub for the Mac, but use another GUI if you prefer.[3]

Download GitHub from http://mac.github.com/, and drag into your application folder, and perhaps your Dock.

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 all 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

[3] Another GUI is git gui which comes with git; type the command `git gui` in a terminal window.

collaboration over the internet.

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 GitHub GUI

- Initialise an existing directory for git by dragging the folder icon onto the GitHub application or its window. Click `Yes`.
- In `Repositories` view, double click on the entry for `myproject`.
- Click on `Settings` in the left-hand tabs: then type lines `.gitignore` and `.DS_Store` into the `Ignored files` window and click `Save Changes` on bottom-right.[5]
- To add files to be managed, click on the `Changes` tab on the left: by default GitHub adds all files to management, change if you wish; files

[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.

[5] You may also want to ignore other files including `*.aux`, `*.log`, `*.out`, `*.synctex.gz` and `*.blg`. The asterisk matches all files with that extension.

managed but unchanged are not shown; the contents of shown files are displayed in the right-hand pane.

- To commit: enter a message such as "Initial version" in the single line in the top-left, and enter more detail if you wish just below; and finally click `Commit Changes` button near top-left.

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 GitHub GUI

- Work, edit and refine files in your project as usual.
- Start GitHub and double-click the project directory from the list in the `Repositories` window.
- Managed files you have edited will be listed; unmanaged files are also listed; managed files that have not changed are not listed.

- Optional, click on the file name in the left-pane to see, in the right pane, the `diff`erences between the current version and the last commit.
- By default, all new files (even in a new directory) and all changed files will get committed; change if you wish.
- To commit: enter a message such as "Initial version" in the single line in the top-left, and more detail if you wish just below; and finally click `Commit Changes` button near top-left.

2.3 Review your work

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

In the GitHub GUI, just click on the `History` tab on the top-left.

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

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

`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.

In GitHub's history view, click `Rollback to this commit` button.

`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

To share and collaborate we need an information store on the web. The company GiHub provides some free storage (as well as a commercial service). To *start*, let's use GitHub.

Create an account Go to https://github.com/plans and click on `Create a free account`. Follow the instructions which involves: registering; generating an ssh key to give to GitHub; saving the API-token from GitHub into your git preference; and setting git config. GitHub's instructions lead you through the process.

3.1 Clone a collaborative existing project

Assume that a collaborator, say `username`, has established a repository, say `ourproject`: they must give you collaboration rights by web browsing to the repository, clicking `Admin`, clicking `Collaborators` on the left menu, then entering your GitHub name and `Add`ing.

- Use a web browser to connect to GitHub https://github.com, and login (top-right).
- Find the project (somehow): for example, just go to https://github.com/username/ourproject
- On the left-side beneath "username/ourproject" and beneath "Code", you should see and click on `Clone in Mac`.
- Click `OK` and then choose a location on your computer for the repository folder to be created and material copied.
- Drag that new folder onto GitHub for GitHub to manage the local repository.

- Thereafter, work and commit as in Sections 2.2–2.3, but additionally occasionally *Click the* `Sync` *button on the* `Changes` *window.*[6]
- Checking and downloading, 'pulling', any changes by your collaborators is one additional step you might like to do before starting work each session: from menu `Repository` select `Pull`.

3.2 Create a new external repository

Yet to do.

[6] I believe the `Sync` operation not only copies information to the remote repository, but also merges into your local repository any changes made there by collaborators in parallel to you.