Version Control with Git

Wolfman and Dracula have been hired by Universal Missions (a space services spinoff from Euphoric State University) to investigate if it is possible to send their next planetary lander to Mars. They want to be able to work on the plans at the same time, but they have run into problems doing this in the past. If they take turns, each one will spend a lot of time waiting for the other to finish, but if they work on their own copies and email changes back and forth things will be lost, overwritten, or duplicated.

A colleague suggests using version control (reference.html#version-control) to manage their work. Version control is better than mailing files back and forth:

- Nothing that is committed to version control is ever lost. Since all old versions of files are saved, it's always possible to go back in time to see exactly who wrote what on a particular day, or what version of a program was used to generate a particular set of results.
- As we have this record of who made what changes when, we know who to ask if we have questions later on, and, if needed it, revert to a previous version, much like the "undo" feature in an editor.
- When several people collaborate in the same project, it's possible to accidentally overlook or overwrite someone's changes: the version control system automatically notifies users whenever there's a conflict between one person's work and another's.

Teams are not the only ones to benefit from version control: lone researchers can benefit immensely. Keeping a record of what was changed, when, and why is extremely useful for all researchers if they ever need to come back to the project later on (e.g., a year later, when memory has faded).

Version control is the lab notebook of the digital world: it's what professionals use to keep track of what they've done and to collaborate with other people. Every large software development project relies on it, and most programmers use it for their small jobs as well. And it isn't just for software: books, papers, small data sets, and anything that changes over time or needs to be shared can and should be stored in a version control system.

Prerequisites

In this lesson we use Git from the Unix Shell. Some previous experience with the shell is expected, *but isn't mandatory*.

Getting ready

Nothing to do: you're ready to go!

Topics

1. Automated Version Control (01-basics.html)
2. Setting Up Git (02-setup.html)
3. Creating a Repository (03-create.html)
4. Tracking Changes (04-changes.html)
5. Exploring History (05-history.html)
6. Ignoring Things (06-ignore.html)
7. Remotes in GitHub (07-github.html)
8. Collaborating (08-collab.html)
9. Conflicts (09-conflict.html)
10. Open Science (10-open.html)
11. Licensing (11-licensing.html)
12. Hosting (12-hosting.html)

Other Resources

- Reference (reference.html)
- Discussion (discussion.html)
- Instructor's Guide (instructors.html)
