This is an electronic version (postprint) of an article published in *Behavioral and Social Sciences Librarian*. The article is available online at http://www.tandfonline.com/doi/full/10.1080/01639269.2015.1062586 (subscription required).

Full citation:
Davis, R. "Git and GitHub for Librarians." Behavioral & Social Sciences Librarian 34.3 (2015): 159–164. Web.

_____

***Internet Connection***

# Git and GitHub for Librarians

Robin Camille Davis

*Abstract: One of the fastest-growing professional social networks is GitHub, an online space to share code. GitHub is based on free and open-source software called Git, a version control system used in many digital projects, from library websites to government data portals to scientific research. For projects that involve developing code and collaborating with others, Git is an invaluable tool; it also creates a backup system and structured documentation. In this article, we examine version control, the particulars of Git, the burgeoning social network of GitHub, and how Git can be an archival tool.*

One of the fastest-growing professional social networks is GitHub, an online space to share code. GitHub is based on a version control system called Git, used in many digital projects that involve developing code, working with others, and creating documentation at every step. For librarians, learning how Git works and how to navigate GitHub are skills useful in contributing to and supporting tech projects and scholarly research.

## Version control in a nutshell

Version control is a concept close to the hearts of many librarians and technologists. With each new version of a digital project, a simple snapshot is taken and an entry is added to the change log, documenting what was changed. The record of the project therefore includes all of its iterations, like a scrapbook of a growing baby complete with dates, author attributions, and useful captions. The code and its documentation are all stored in the project's *code repository*, the "official" version of the project. Librarians are fans of version control systems because they produce rich but lightweight metadata at every step of the project.

Most digital projects use version control (or should). Version control is equally adept at managing one individual's personal web project or a large open-

source code project written by many collaborators. One of the most popular version control systems is Git, free and open-source software available to download from git-scm.com. (Another popular version control system is Subversion, or SVN, but it has waned in popularity recently.) Once downloaded, Git is operated from the command line interface. On a Mac or Linux, that would be the built-in application called Terminal; on Windows, it is called Command Prompt. Git interacts with the files in a certain folder. It operates in the background as a sort of shadow system; once you decide to use Git in a folder, it creates hidden files in that folder containing its logs and other metadata.

Git users typically take on their own style and conventions. Some use just the basic system components; some use all the bells and whistles. Here, we will walk through several simple workflows to demonstrate the use of the software.

**Git workflows**

A simple use of Git would involve a *local* project (kept on a person's computer). Designating some files as being part of the *repository*, the person can keep a log of all changes to these files by *committing* (logging the change) and *pushing* (finalizing changes). Pushing code is the culmination of a number of steps: making changes, logging them, and selecting which of the changed files are ready to go live. Code changes are committed and pushed iteratively. The end of a long project may have hundreds of commits logged.

An example of this workflow could be a programmer working on a small program. As she tries implementing different parts of her code, she wants to make sure she can retrace her steps. She might realize she has made a few unsatisfactory changes, and revert to a previous version of the code, an easy task in Git. Then she might find a good solution and commit/push it, signifying a final (for now) version for herself.

[image: workflow-small.tif]
*A small project workflow using Git*

One common website workflow could involve two instances of the website, and therefore two local repositories: one for internal testing (*development*), and one for live public view (*production*). Between these two instances is a *remote repository*. A website administrator will make a number of changes on the development website, for example, to implement a new search box. She tests out the new search box on the development server, committing the changes she makes. Once she is happy with how the updated development website functions with the new search box, she will push the updated code to the remote repository, then *pull* the code to the live (production) server. The remote repository is the middleman, the master codekeeper that is offsite. A repository is remote for two reasons: one, so that it functions as an offsite backup of the code; and two, because version control services like GitHub provide many other features, which we shall examine shortly. Pulling the code from the remote repository to the live site simply updates the site with code that the website administrator has already tested and documented.

[image: workflow-website.tif]
*One model of a website workflow using Git*

Finally, another view of a Git workflow is as a collaborative tool, where Git truly shines. Each collaborator has her own local repository on her machine. She makes changes to her part of the project and pushes it to the remote repository. Periodically, she will pull updated code from the remote repository to her local one, ensuring that she's working on the most up-to-date files. Git will *merge* all the changes from all collaborators, elegantly stitching all the updated code together, even updates made to the same file. (If two collaborators try to change the same thing in the same file, Git will alert them.) Git makes collaborative work much easier. Rather than trying to finagle which version of which file someone should be working on, Git takes the frustration out of the situation by updating everyone's code to the current version while ensuring that no one's work is overwritten.

[image: workflow-collab.tif]
*One collaborative workflow using Git*

The following is a short example of what Git looks like for the user on the command line.

```
1  robin@Library: $ git commit -m "Date now displays on blog posts"
2  [master c8f7610] Date now displays on blog posts
3  1 file changed, 1 insertion(+), 1 deletion(-)
```

Line 1 is the user's command signifying that she wants to finalize a given file with the log message, "Date now displays on blog posts." Line 2 is the response from the Git system, displaying the commit number (ID # c8f7610) along with the log message she just recorded. Line 3 tells the user that one file was changed, with one character inserted and deleted. With this example, we see that Git is terse but elegant. Once the user masters the basic commands, she can run the workflows previously mentioned.

**GitHub, a code-sharing social network**

Again, Git is a version control system. It's free and open-source software that can manage files locally and remotely. GitHub, on the other hand, is a private company that facilitates sharing code via the Git system. The company characterizes itself thus: "GitHub is the best place to share code with friends, co-workers, classmates, and complete strangers. Over eight million people use GitHub to build amazing things together … GitHub has grown into the world's largest code host" (GitHub).

GitHub is a functional remote repository. Public accounts are free; private accounts cost a monthly fee. But GitHub is more than a practical part of one's workflow. A public GitHub account displays information about the person, all of her repositories, and all of the files inside each public repository. In short, a person can make all of her code publicly viewable. She can view others' code, too. She can download and modify someone else's code for her own purposes (known as *forking*

code). Moreover, she can contribute her revisions to others' code (a *pull request*), a feature particularly useful for wide-scale, open-source collaborative projects.

GitHub has thus become an active social network in and of itself, where people with shared interests can follow each other and see the progress on various projects. The open nature of the site facilitates collaborations and connections. An interested viewer can watch (sign up for notifications of changes), star (mark as favorite), and fork a repository; these metrics are like citation counts for code. For programmers, GitHub accounts function as résumés, too, complete with these built-in commendations from their peers (Begel, Bosch, & Storey 2013).

In addition to individual accounts, GitHub offers organizational accounts. Libraries with an active presence on GitHub include the Digital Public Library of America (username @dpla), the New York Public Library (@nypl), the Library of Congress (@LibraryOfCongress), as well as many university libraries, including the University of Arizona (@ualibraries, which houses the popular Guide on the Side software) and Stanford University (@sul-dlss, which includes Spotlight, a digital exhibit site builder). Browsing libraries' and librarians' GitHub accounts, one can view some of the latest code developments and explore how popular projects have come to fruition. (Note that repositories typically share code, but not large data sets or content collections. Git is designed for tracking files that change frequently throughout development, mostly code. While a digital project may be designed to work with big data or image collections, those files don't have to be tracked by Git and are therefore not accessible in GitHub.)

Beyond libraries, other organizations of interest include many government agencies. These are hundreds in number and include Data.gov.uk (@datagovuk, which includes code for the UK's government data portal); the city of Helsinki (@City-of-Helsinki, which provides an API for government documents); and accounts from smaller government offices, too, like the Idaho Department of Fish and Game (@idahofishgame, which includes the back-end code for an educational bird program).

**GitHub and scholarly work**

At a time when Open Access and open data have gained incredible momentum, GitHub has become a site for academic transparency. It provides a well-structured place to post code and lightweight data used in a scholarly endeavor, encouraging reproduction of results and reuse in other projects. For instance, statistical code, text mining scripts, and custom-built software used in the course of scholarly research could all be shared openly on GitHub. Moreover, Git's built-in documentary features mean that the entire history of the project could be publicly viewable and searchable, including the change log with its timestamps and author attributions (Ram 2013).

Scholarly output can include a final publication and code, but there are more opportunities for openness. Git is not limited to tracking code: any file can be tracked in Git, including documents. Thus, many writers are finding that structured version control is much more robust and flexible than the "track changes" option in Microsoft Word. Scholarly research can therefore be wholly tracked in Git and made available on GitHub from start to finish, from initial drafts and messy code to the

final manuscript and finished software. In fact, entire books have been written on GitHub, like Samy Pessé's popular nine-chapter *How to Make a Computer Operating System in C++*. In traditional citation, Pessé is the sole author. On GitHub, however, the book has 34 contributors — mostly users editing typos or requesting clearer explanations.

A discerning scholar will likely not want *all* of her work publicly available, particularly if private data or other ethical questions are involved. Librarians working with researchers may encourage enthusiastic but careful use of GitHub.

**GitHub, Git, and archives**

GitHub has been online since 2008, but web users have seen enough large-scale networks wither and die to know that they can't count on GitHub to last forever, and GitHub has made no such claim. Thus archives and scholarly repositories may be the final homes of Git-powered projects in a post-GitHub world.

Today's archivists find clues about a manuscript's provenance by examining marginal notes and creating a genealogy of previous and subsequent versions; tomorrow's digital archivists have only to look at a project's structured Git documentation to find which author wrote any given line of a file and when. Admittedly, digital preservation on the whole is notoriously difficult. But should a Git repository become an archival object, its entire history would be laid bare, the equivalent of every version of an author's manuscript bundled into one searchable digital item. Digital preservationists view version control as one potential tool in ascertaining an object's "chain of custody" (Five Colleges Consortium n.d.). Finished projects can preserve Git's functionality if the file hierarchy is preserved and Git's hidden files are included in the archival object. Human-readable documentation outlining the project's use of Git, as well as any permissions details, would assist the archival process as well.

Git is a librarian's dream tool. It is metadata come to life — metadata that powers projects that power cities, metadata that connects people to each other, and metadata that is lightweight yet impressively granular. Learning version control is useful for personal or organizational projects and for advising researchers, and it is a key skill for the libraries and archives of tomorrow.

References

Begel, A., J. Bosch, and M.-A. Storey. "Social Networking Meets Software Development: Perspectives from GitHub, MSDN, Stack Exchange, and TopCoder." *Software, IEEE* 30, no. 1 (2013): 52–66. doi:10.1109/MS.2013.13.

Five College Consortium. "Digital Preservation: A Planning Guide for the Five Colleges," n.d. Accessed May 20, 2015. https://www.fivecolleges.edu/libraries/digital-preservation/digital-preservation-a-guide-for-the-five-colleges.

Github. "GitHub." *GitHub*. Accessed May 20, 2015. https://github.com.

Pessé, Samy. *How to Make a Computer Operating System in C++*. GitHub, n.d. Accessed May 20, 2015. https://github.com/SamyPesse/How-to-Make-a-Computer-Operating-System.

Ram, Karthik. "Git Can Facilitate Greater Reproducibility and Increased Transparency in Science." *Source Code for Biology & Medicine* 8, no. 1 (May 2013): 1–8. doi:10.1186/1751-0473-8-7.