Git and GitHub: A Short Introduction

Michael Assmus

10/29/2017

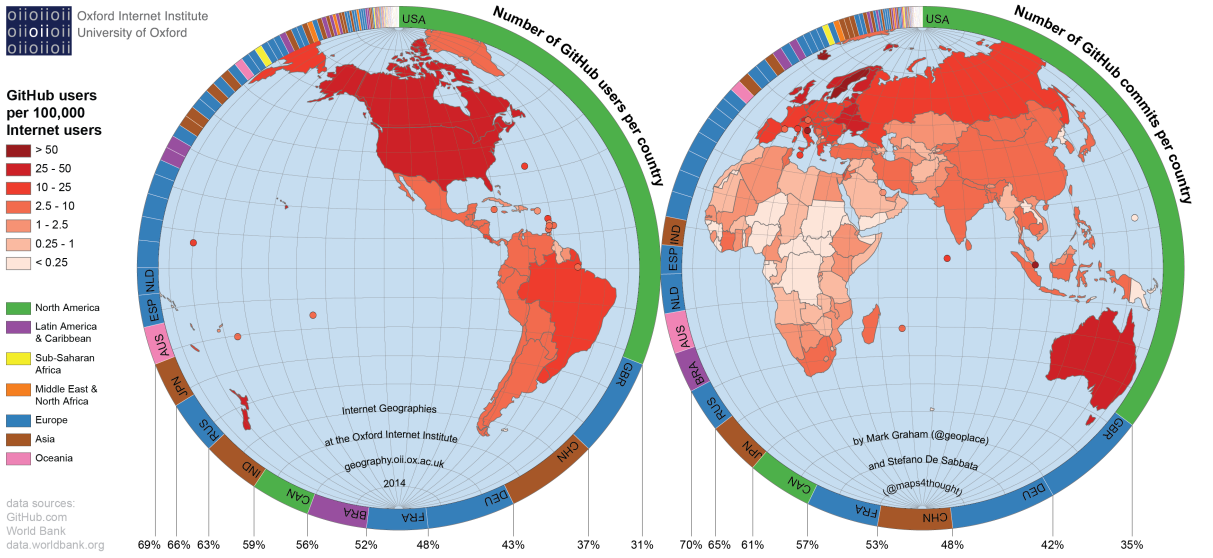
Introduction

For this assignment, I wanted to look more into the historical background of Git and GitHub, as well as looking a bit more closely at how they work and what they do. I'll start off with the historical context, and then expand on the other topics in turn.

History

In 2005, relations broke down between the Linux kernel development community and BitMover, the company that had created the proprietary version control software known as BitKeeper. Three years earlier, in 2002, the kernel developers had chosen to move from the traditional storage and editing method, where files and patches were passed around from one person to another, to a digital version control system, which would let them keep better track of changes, share more effectively, and revert to old versions as necessary. Their chosen system, BitKeeper, at the time could be used for free as long as users agreed to not create any competing version control software. Unfortunately, in 2005, one of the Linux kernel developers created software that the BitKeeper staff saw as being in breach of that contract, and so BitMover rescinded the free use of BitKeeper by the kernel development community. While this led the Linux developers, led by Linus Torvald, the creator of Linux, to migrate their data off the BitKeeper system, the developers still favored the use of a version control system if one was available. Torvald, digital inventor that he was, chose to create his own, and thus Git was born.

However, while Git was very useful to expert programmers and Linux users like Torvald, it was rather unapproachable for everyone else. As such, some years later in late 2007, GitHub was created to solve this problem. It created a relatively simple and accessible website based on Git architecture that coders could use to store, share, and manipulate code, text, and other sorts of files. Free of charge to anyone willing to share their repositories publicly, GitHub has risen rapidly in popularity, and now boasts millions of users from all over the world.



GitHub worldwide users in 2014

Purpose

Git was built by the Linux kernel development community as a replacement for the BitKeeper version control system. It incorporated lessons they had learned while working with BitKeeper, and was designed to be fast, simple, capable of working with very large projects like the Linux kernel, and able to be used for mass parallel development (where many developers could work on parts of the code at the same time without issues).

GitHub is a website and an easily-accessible online storage and sharing space for Git repositories. It was created with the goal of making Git usable by a much larger audience, providing a relatively simple interface and a great deal of accessible utility.

Judging by their popularity, both Git and GitHub fulfilled their purposes handily.

Functionality: Git

Git, as mentioned in the history section, is a version control system, which means it tracks changes in selected files and can be used to retrieve or reference past versions of those files. Git is somewhat unusual in its methodology, however; whereas most version control systems record a set of selected files and the changes to those individual files over time, Git instead takes "snapshots" of the entire tracked file system at various points in time. In other words, instead of storing a "series of changes", Git stores a "stream of snapshots".

This method provides a number of advantages. Most prominently, storing the information as a series of snapshots makes it very easy to "branch" the data, splitting off from the main line of development and making changes without affecting the primary files. This is very useful when a user is trying to experiment with and test certain changes or additions without having to worry about reverting to previous versions if something breaks, and it also makes it much easier for multiple people to collaborate on a project, since they can each work on their own branch and then push their changes to the project administrator for review and possible implementation.

The most common way to interact with Git is to use your computer's command line interface. There are various programs used to access the command line on different operating systems, such as Terminal on Macintosh and Command Prompt and Powershell on Windows. Generally

speaking, commands for git follow the following format:

`$ git <command> <additional variables>`

The most commonly-used commands are generally `$ git status`, `$ git add`, `$ git commit`, and `$ git push`. The “status” command checks whether the files in the current directory are being tracked by Git, and of those files that are being tracked, it determines whether they’ve been altered since the last time Git took a snapshot of them. The “add” command lets you select untracked files and files that have been recently changed and add them to a commit. The “commit” command takes a new copy of all files added to the commit using the “add” command, saving a newly updated snapshot of the filesystem to its permanent records for later retrieval. The “push” command updates a remote directory using the local directory, uploading new snapshots and data to a repository like GitHub and thereby allowing you to secure your data and share it with other developers and users.

Functionality: GitHub

GitHub acts as an online repository for Git users. People with GitHub accounts can use the “push” command to push copies of their local Git repositories to GitHub, and then they can easily look at the version histories of tracked files and share their uploaded files with other GitHub users. GitHub also offers direct editing of files, as well as the ability to fork off a new branch of another user’s file or repository. Such branches can be kept and modified for personal use, but they also provide the ability to push changes back to the original repository, which the owner of that repository can then look over and possibly accept as an edit to their “master” file repository.

GitHub also offers a variety of other utilities, such as the ability to specify collaborators (people with greater access, including push access) for a given repository, the ability to create a project “wiki” that lays out project information and plans, the ability to subscribe to particular repositories in order to see when updates are made, the ability to search for public repositories by keyword, and many other options.

Benefits

While many of the benefits of Git and GitHub should be obvious from the above descriptions and from personal use, it’s worth summarizing some of them here.
- Security: As a version control system, Git allows you to easily save older versions of your files and return to those versions if necessary. GitHub acts as additional insurance by saving copies of your files and their past versions on a remote server, which can be accessed even if something goes wrong with your local versions.
- Collaborative development: By way of easy branching, Git and GitHub make it easy for many people to work on a single project, making changes on their own versions and then pushing them to a master version.
- Experimentation: Users can safely experiment with new code ideas, making and testing changes to files without needing to worry about accidentally wrecking things or losing the working version. If something goes wrong, the master version is safe.
- Sharing: Through GitHub, people can easily share many different types of files with the public, allowing them to distribute code, text, images, and other types of information.
- Price: Git is entirely free to use. GitHub only charges users if they want to keep repositories private; if they stick to public repositories, GitHub is free of charge.

Conclusion

In summary, Git was born out of necessity, when a community of expert developers needed a new version control system that was easy, fast, effective, and free. GitHub, in turn, was created to bring Git to a wider audience, providing a remote repository that was simple and easily accessible to the public. Together, Git and GitHub provide peace of mind to the solo developer, while offering easy mass collaboration to those who are working in groups. While not necessarily ideal for every project, Git and GitHub are of great utility in a wide variety of circumstances, and are strongly recommended for novice programmers and coding veterans alike.