Why we need Git?

Hamid Semiyari

Git

- Git is the most popular `version conrol` system in the world. A vesrion control system records the changes that made to our code over time in a special database called `repository`. We can look at our project history and see who has made what changes when and why. If we made a mistake on something we can easily revert our project back to an earlier state.
  In summary, with version control system we can track history and work together.

- Git is free, open source, super fast, and scalable. Operation such as `branching` and `merging` are slow in other version control system like `subversion` or `tfs`But they are very fast in Git.
- Git is everywhere, almost every job description (programming related) mentions Git.

Version Control:

- **Version control system**: a program that tracks iterative changes of files. Git is the most popular version control system.
- You can go back to previous versions of your code/text, then move forward to the most recent version, or keep the old version.
- You can create copies of the code, change them, then merge these copies together later.

Motivation 1: Change code without the fear of breaking it

- You want to try out something new, but you aren't sure if it will work.
- Non-git solution: Copy the files

  - analysis.R,
  - analysis2.R,
  - analaysis3.R,
  - analysis_final.R,
  - analysis_final_final.R,
  - analysis_absolute_final.R,
  - analysis7.R
  - analysis8.R

- Issues:

  - Difficult to remember differences of files.
  - Which files produced specific results?

- Git lets you change files, keeping track of old versions, and reverting to old versions if you decide the new changes don't work.

Motivation 2: Easy Collaboration

- In a group setting, your collaborators might suggest how to change your analysis/code.
- First non-git solution: Email files back/forth.
- Issues:

  - You have to manually incorporate changes.
  - Only one person can work on the code at a time (otherwise multiple changes might be incompatible).

- Second non-git solution: Share a Dropbox or Google Docs folder (a "centralized" version control system).
- Issues:

  - Again, only one person can work on the code at a time.
  - Less user-friendly for tracking changes.

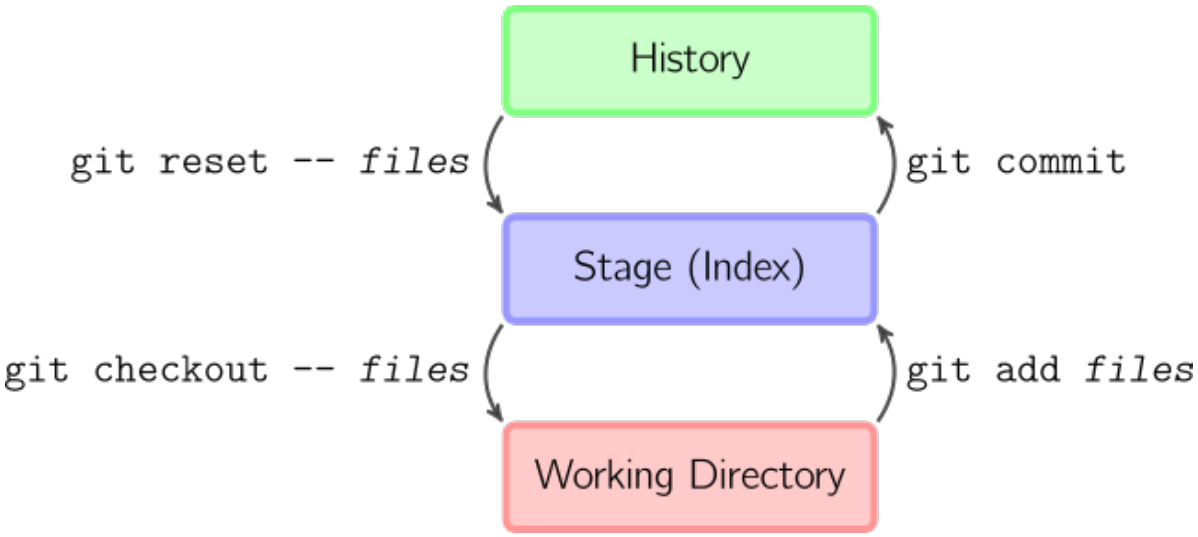
- Git let's each individual work on their own local repository and you can automatically incorporate changes.

Motivation 3: Great for job interviews

- In a 2021 Stack Overflow Survey, 93.43% of developers say they use git., 58.5% of data scientists say they primarily use git for sharing code.
- You can make your final-project repo public so prospective employers can view your work.
- You can host a website on GitHub, increasing your visibility. Doctor Gerard host hosts his personal website and teaching websites on GitHub.

Basic Git

- A **repository** (or repo, for short) is a collection of files (in a folder and its subfolders) that are together under version control. In data analysis, each repository is typically one project (like a data analysis, a homework, or a collection of code that performs a similar task).
- The way git works (graphic from Mark Lodato):



- **Working Directory**: To git, this means the current versions of the files. Changes to files that you haven't recorded only exist in the working directory and are not yet saved in the history.
- **Stage**: Files that are scheduled to be committed to the history, but not yet committed. Only files in the stage will be committed to the history.

- **History**: The timeline of snapshots of files. You commit a file to the history and then, even if you modify it later, you can always go back to that same file version.
- We'll focus on the right-hand-side of this diagram where your workflow is typically:

  1. Modify files in your working directory until you want a snapshot.
  2. Add these modified files to the staging area.
  3. Commit staged files to history, where they will be kept forever.

- The left-hand side of the diagram is used when you want to undo mistakes.
- All git commands begin with `git` followed immediately by an argument for the type of command you want to execute.
- For the right-hand-side of the diagram, the following are the useful git commands:

  - `git init`: Initialize a git repository. *Only do this once per project.*
  - `git status`: Show which files are staged in your working directory, and which are modified but not staged.
  - `git add`: Add modified files from your working directory to the stage.
  - `git diff`: Look at how files in the working directory have been modified.
  - `git diff --staged`: Look at how files in the stage have been modified.
  - `git commit -m "[descriptive message]"`: commit your staged content as a new commit snapshot.

How to use Git?

- Command line: We can use Git on the command line. Open a termianl or command window to execuate git commands. This the fastest sometimes easiest way to get the job done.
- Code editors & IDEs (or Integrated Development Environment): Most code editors and IDEs have built-in support for basic Git features.
- Graphical User Interfaces: There are GUIs specifically made for using Git. You can find the complete list of these tools for different platforms on the Git website. Click here. GUI tools are not always available. GuI tools have somelimitations. You might connect to a server remotly and you may not have permission to install GUI tool. So to my opinion it is better to use commond line. There are many people use both GUI tool and command line.
- A code editor is a text editor that has some features that make writing code easier. For instance, code editors will automatically highlight words based on syntax and will automatically indent lines of code correctly. An IDE is usually more complex than a simple text or code editor. It has more features, it takes time to learn, but in the end is more powerful. It is a piece of software that allows developers to create, modify, and

debug code easily. A code editor doesn't have the debugging and auto-complete features that an IDE has. Atom is a code editor and Rstudio is an IDE.