An Introduction to Git

https://xkcd.com/1597/

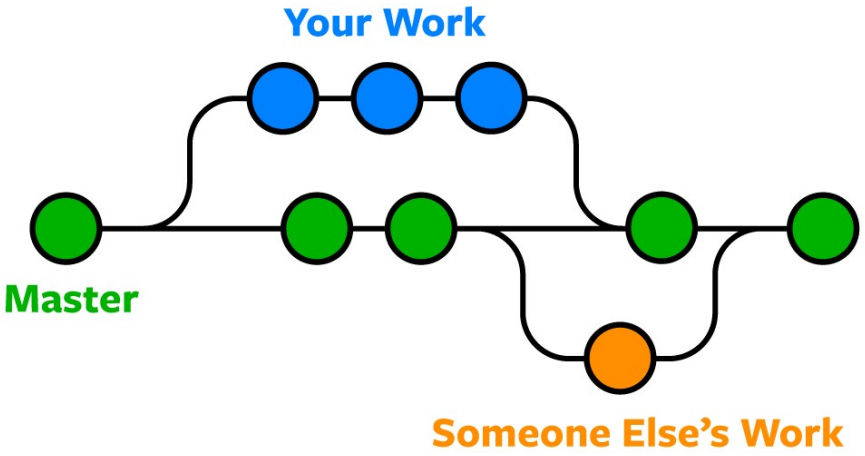
Source/Version Control

- Collaboratively track changes in a series of files using a centralized storage system
- Many different flavors, each slightly different
  - Git, SVN, Mercurial, Helix Core, TFS, ClearCase
- Shared characteristics
  - Centralized storage
  - Track who made what change, when, and why
  - Work in parallel on the same set of files

Basic Git Nomenclature[1]

[1] https://guides.github.com/introduction/git-handbook/

TERM	DEFINITION
Repository	A collection of software and/or files for collaborative editing
Local	Location where you and the computer you are working on are
Remote	Location that is accessible to you and project collaborators, e.g. Github
Client	Your local computer
Clone	Create a local copy of a remote repository
Commit	Save your changes – Locally for Git
Branch	A parallel series of changes starting from some point in history
Merge	Combine changes from one branch with another
Pull	Get new commits from the remote to your local client
Push	Send local commits from your client to the server
Pull Request	Process to review and make comments on a merge

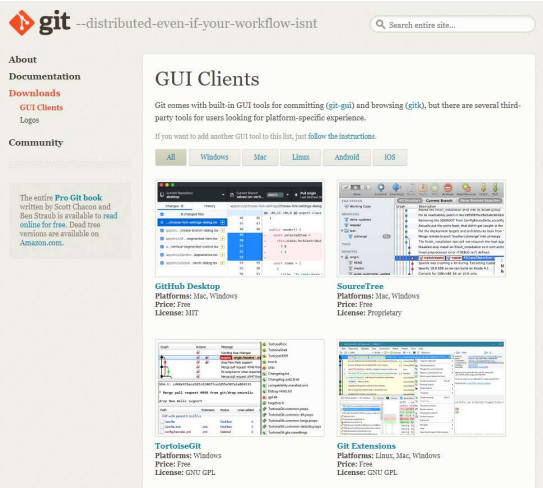


Git vs. Github

- [1] https://github.com/join
- [2] https://git-scm.com/downloads

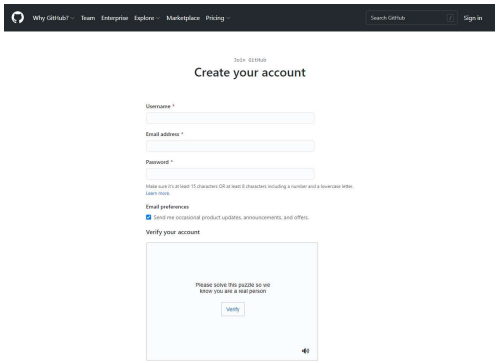
Git

- Protocol for interacting between the client and server

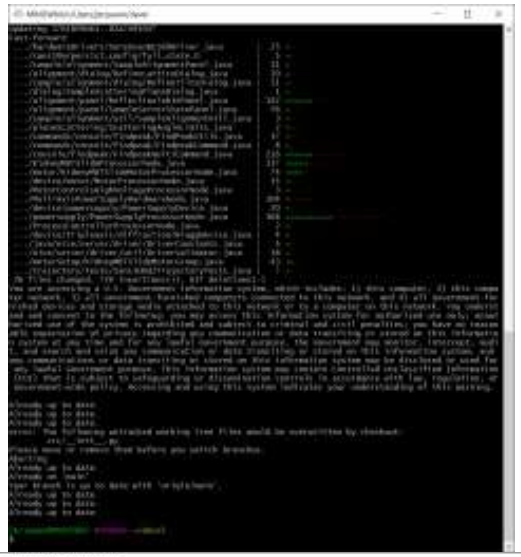


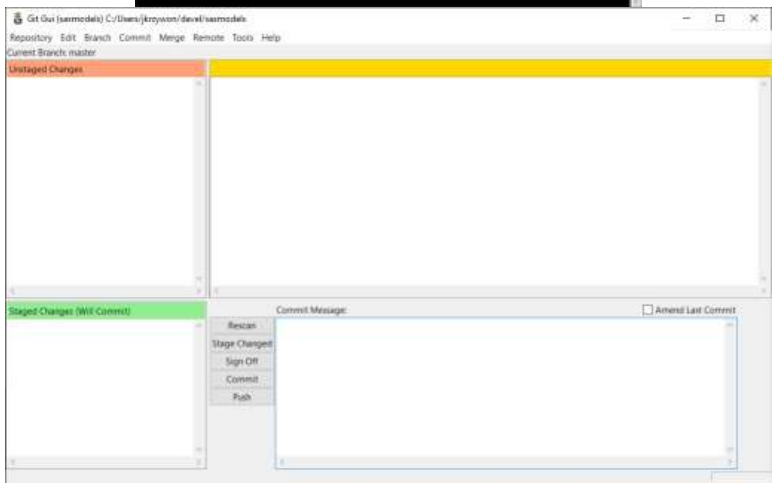
Github

- One of many hosts that uses the Git protocol
- Other hosts include Gitlab and Bitbucket

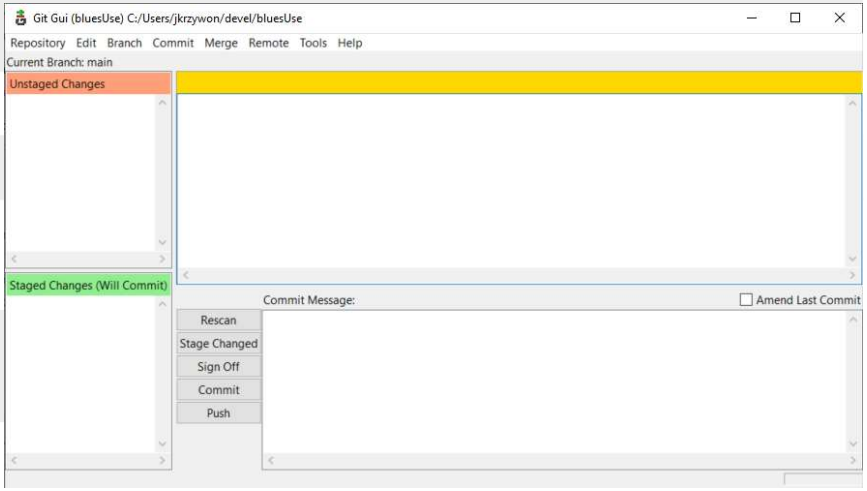


Git GUI vs. Git Command Line Interface (CLI)?



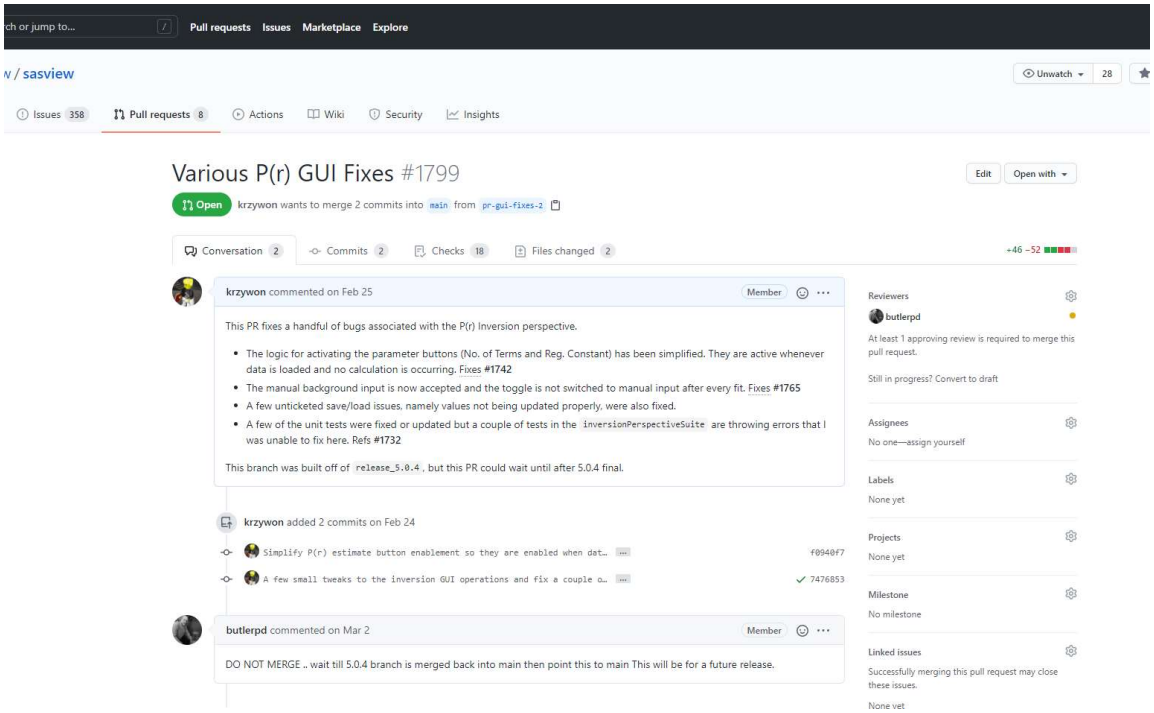


Usage

Command	Git CLI	Git GUI
Commit	git add <file1> <file2> <etc.> git commit –m “<commit message>”	
Pull	git pull	
Push	git push –u origin <local_branch_name:remote_branch_name>	
Change Branch	git checkout <branch_name>	
Create Branch	git branch <branch_name> - Create new branch, stay in current git checkout –b <branch_name> - Create and checkout new branch	
List Branches	git branch	
Delete Branch	git branch –d <branch_name> git branch –D <branch_name>	
Help	git help git help <command_name>	

Pull Requests (PR)

https://github.com/SasView/sasview/pull/1799



- Pull requests should be small and focused for easy review
- Add an itemized list of things being fixed/solved/changed
- Use fixing/closing keywords to link issues to the PR

Good Practices

- Commits should be small and focused
- Commit messages should be informative
- Use ticket references in commit messages (refs #<ticket_number>
- Pull requests should be ready for review - all features present and working locally
- Pull requests should be small for easier review
- Use closing keywords in pull requests (fixes #<ticket_number>

Common Pitfalls

[1] https://www.atlassian.com/git/tutorials/undoing-changes/git-reset
[2] https://www.atlassian.com/git/tutorials/rewriting-history

Pitfall	Potential Solutions
Unintended commits	• git reset HEAD: Throws away local commit(s) that have not been pushed retaining changes made in those commits.[1]
Forgot to add something to commit	• git commit --amend: Amends the last commit made. Can amend the files added/removed, file changes, and/or commit message.[2]
Merge conflicts	• Most editing tools have side-by-side file comparison tool • Possible through CLI, but not easy.
Commit to the wrong branch	• First commit: branch from new commit, then 'git reset HEAD –hard' • Later commit: git cherry-pick – creates a new commit, copying a previous commit contents and message