

An Introduction to Git

<https://xkcd.com/1597/>



Source/Version Control

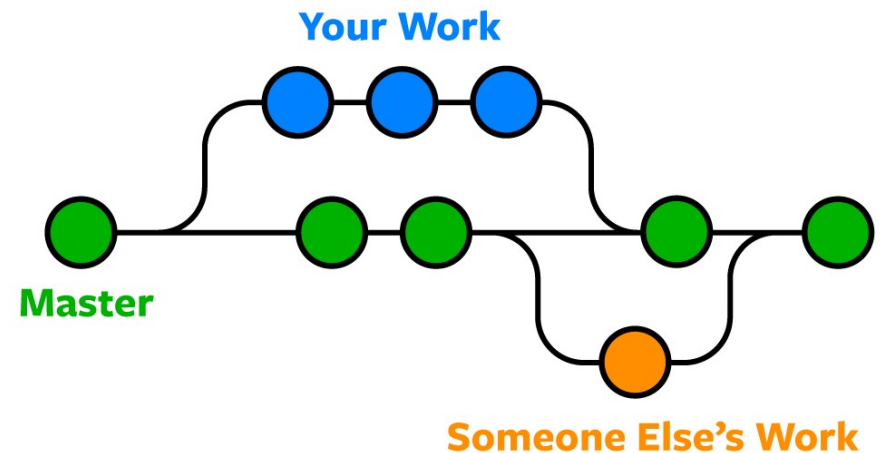
- Collaboratively track changes in a series of files using a centralized storage system
- Many different flavors, each slightly different
 - Git, SVN, Mercurial, Helix Core, TFS, ClearCase
- Shared characteristics
 - Centralized storage
 - Track who made what change, when, and why
 - Work in parallel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



Basic Git Nomenclature¹

[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

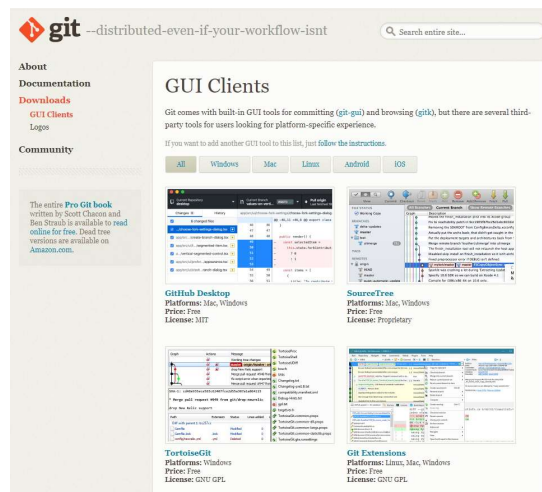


Git vs. Github

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

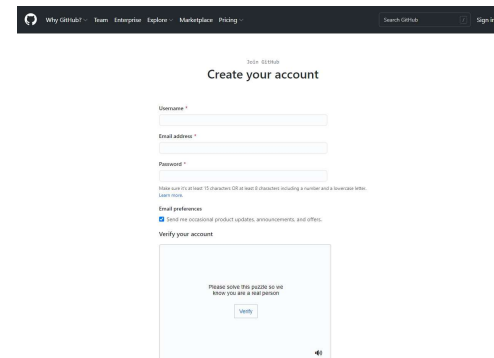
Git

- Protocol for interacting between the client and server



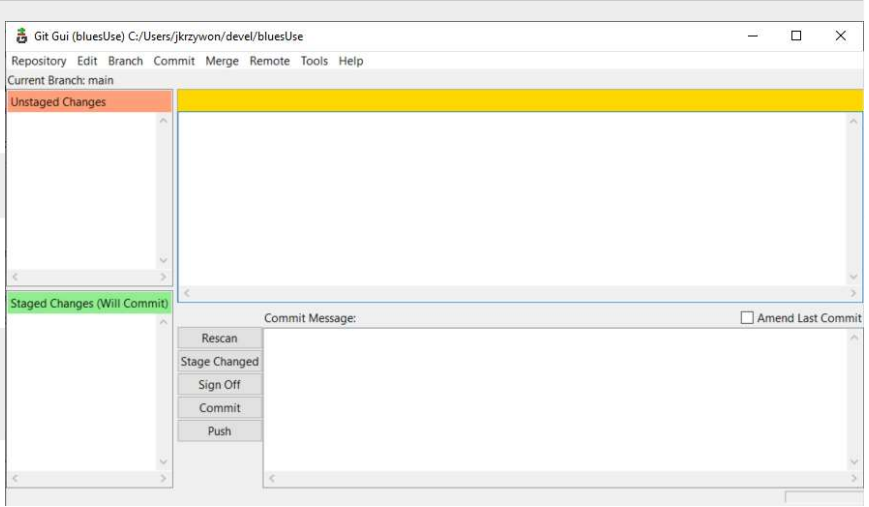
Github

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





Usage

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

Pull Requests (PR)

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

The screenshot shows a GitHub Pull Request (PR) titled "Various P(r) GUI Fixes #1799" in the repository "SasView/sasview". The PR is open and ready for review, with 28 stars and 1799 pull requests in the repository. The PR description states: "This PR fixes a handful of bugs associated with the P(r) Inversion perspective." and lists several fixes: "The logic for activating the parameter buttons (No. of Terms and Reg. Constant) has been simplified. They are active whenever data is loaded and no calculation is occurring. Fixes #1742", "The manual background input is now accepted and the toggle is not switched to manual input after every fit. Fixes #1765", "A few unticketed save/load issues, namely values not being updated properly, were also fixed.", and "A few of the unit tests were fixed or updated but a couple of tests in the 'InversionPerspectiveSuite' are throwing errors that I was unable to fix here. Refs #1732". The PR was created by krzywon on Feb 25 and is currently in progress. The right sidebar shows the review status, with a note from butlerpd: "At least 1 approving review is required to merge this pull request." and "Still in progress? Convert to draft". The bottom section shows the commit history, with two commits added on Feb 24: "Simplify P(r) estimate button enablement so they are enabled when dat..." and "A few small tweaks to the inversion GUI operations and fix a couple o...".

- 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

	COMMENT	DATE
○	CREATED MAIN LOOP & TIMING CONTROL	14 HOURS AG
○	ENABLED CONFIG FILE PARSING	9 HOURS AG
○	MISC BUGFIXES	5 HOURS AG
○	CODE ADDITIONS/EDITS	4 HOURS AG
○	MORE CODE	4 HOURS AG
○	HERE HAVE CODE	4 HOURS AG
○	AAAAAAA	3 HOURS AG
○	ADKFJSLKDFJSDKLFJ	3 HOURS AG
○	MY HANDS ARE TYPING WORDS	2 HOURS AG
○	HAAAAAAAAAANDS	2 HOURS AG

AS A PROJECT DRAGS ON, MY GIT COMMIT MESSAGES GET LESS AND LESS INFORMATIVE.

- 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	<ul style="list-style-type: none">• <code>git reset HEAD</code>: Throws away local commit(s) that have not been pushed retaining changes made in those commits.¹
Forgot to add something to commit	<ul style="list-style-type: none">• <code>git commit --amend</code>: Amends the last commit made. Can amend the files added/removed, file changes, and/or commit message.²
Merge conflicts	<ul style="list-style-type: none">• Most editing tools have side-by-side file comparison tool• Possible through CLI, but not easy.
Commit to the wrong branch	<ul style="list-style-type: none">• First commit: branch from new commit, then <code>'git reset HEAD --hard'</code>• Later commit: <code>git cherry-pick</code> – creates a new commit, copying a previous commit contents and message