

Introduction to Git & GitHub



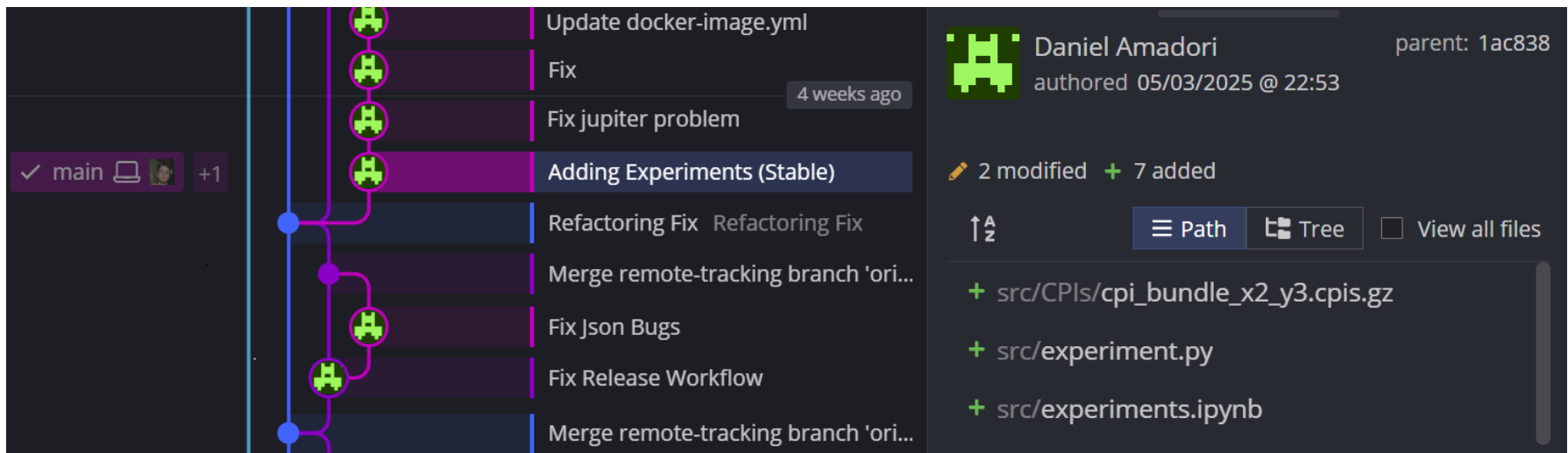
<https://git-scm.com/>



<https://github.com>

Version Control System

- a way to manage files and directories
- track changes over time
- recall previous versions



Git

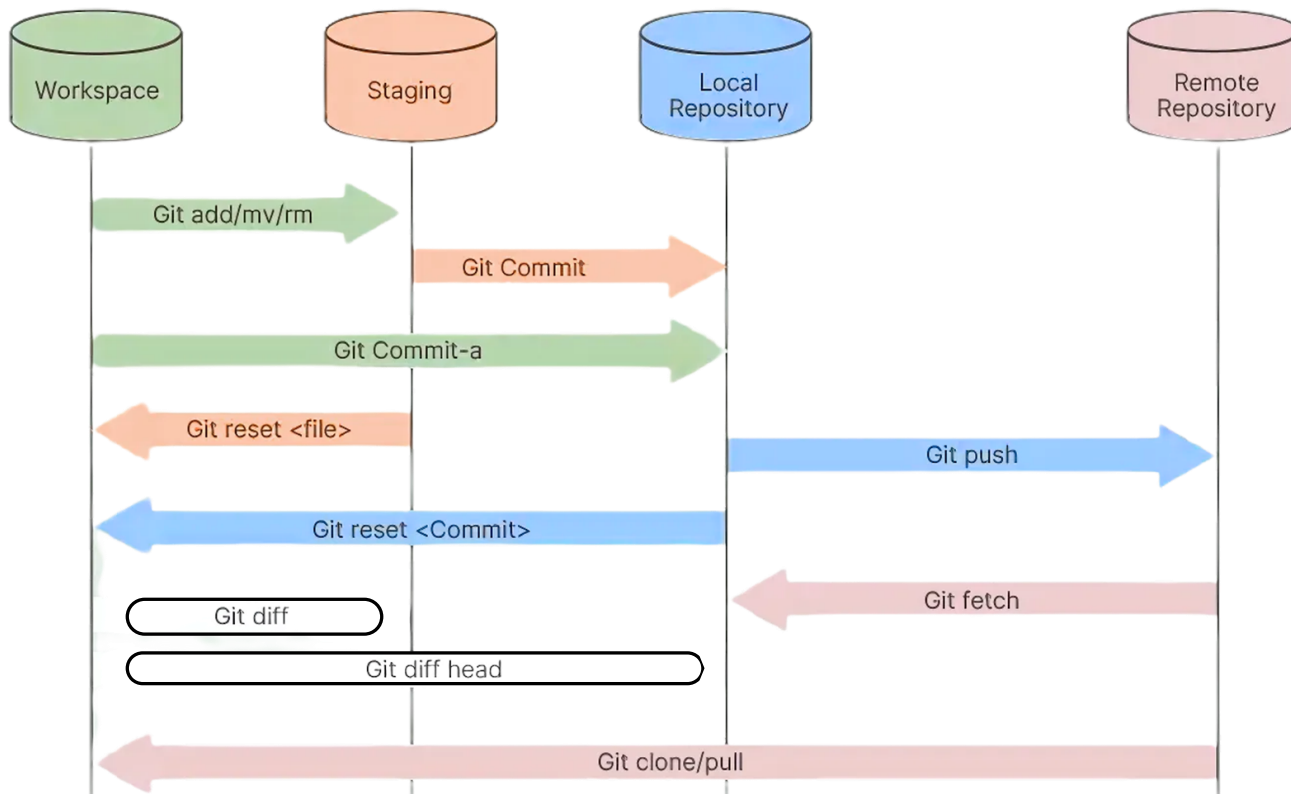
- created by Linus Torvalds, April 2005
- a command line version control program
- uses checksums to ensure data integrity
- cross-platform
- open source, free



Repository

- usually used to organize a single project
- repos can contain:
 - folders, files, images, videos,
 - code and data sets
 - anything your project needs

Git architecture



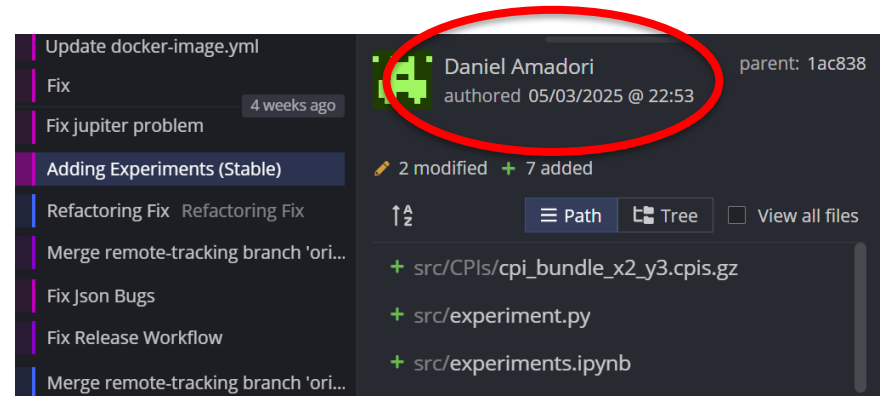
Before starting to use git

- Setup your name and email so others can know who committed changes.
- `git config --global user.name "name"`
- `git config --global user.email "email"`

Note: set for all repositories on your computer

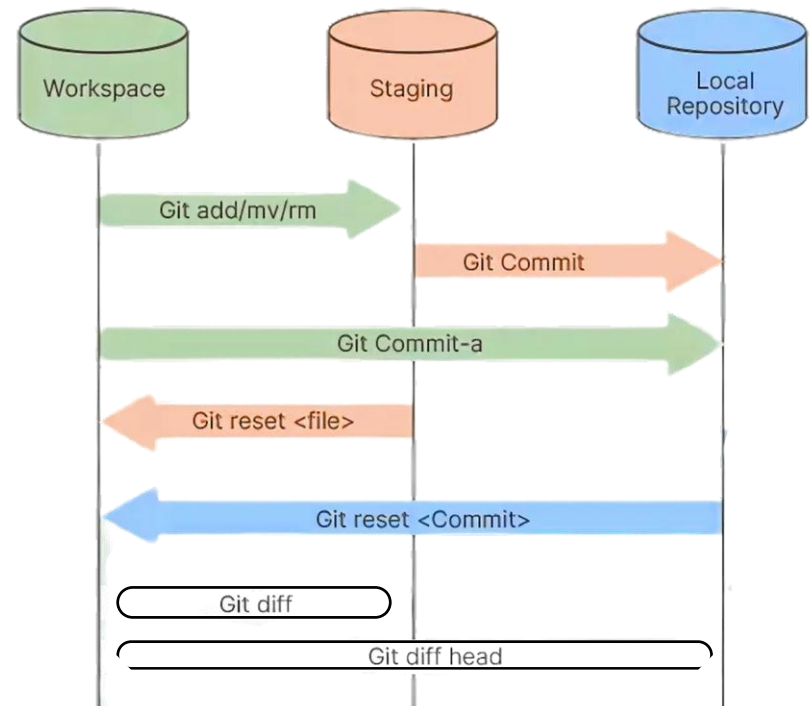
- `git config --local user.email "email"`

Note: can set differently for each repository



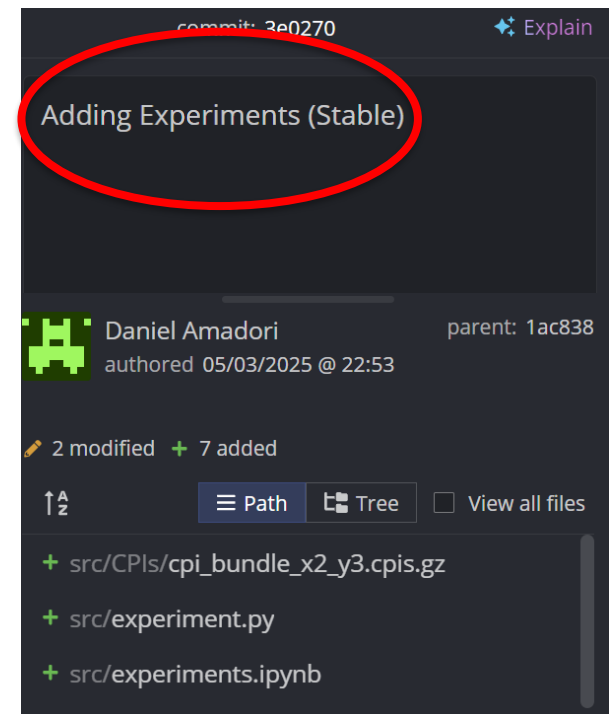
A simple Git workflow

1. Create a Project folder:
`mkdir`
2. Initialize a new project in a directory:
`git init`
3. Create a file:
`touch <filename>`
4. Add in the staged files:
`git add <filename>`
5. Commit the change to the repo:
`git commit -m "important message here"`
6. Watch changes:
`git log`



Commit messages

- Tell what it does (present tense)
- Single line summary followed by blank space followed by more complete description
- Short summary of changes
- More detailed explanatory text, if necessary.

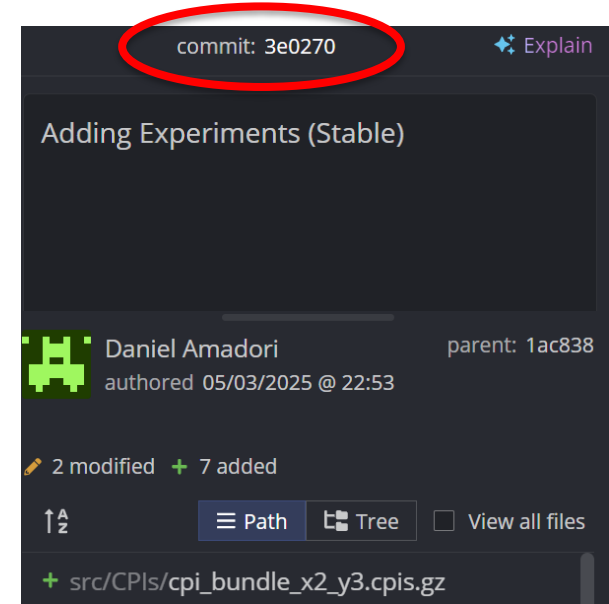


SHAs

- Checksums generated by SHA1 encryption algorithm

The HEAD pointer

- points to a specific commit in repo
- as new commits are made, the pointer changes
- HEAD always refers to the latest commit on the currently active branch.



Basic Git Commands Overview

- See where files are in the three tree scheme
git status

- Committing and adding message

git commit -a

- Allows one to add to staging index and commit at the same time
- Grabs everything in working directory
- Files not tracked or being deleted are not included

- Compares changes to files between repo and working directory

git diff

- Deleting files from the repo

git rm <filename>

moves deleted file change to staging area

It is not enough to delete the file in your working directory. You must commit the change.

- Moving (renaming) files

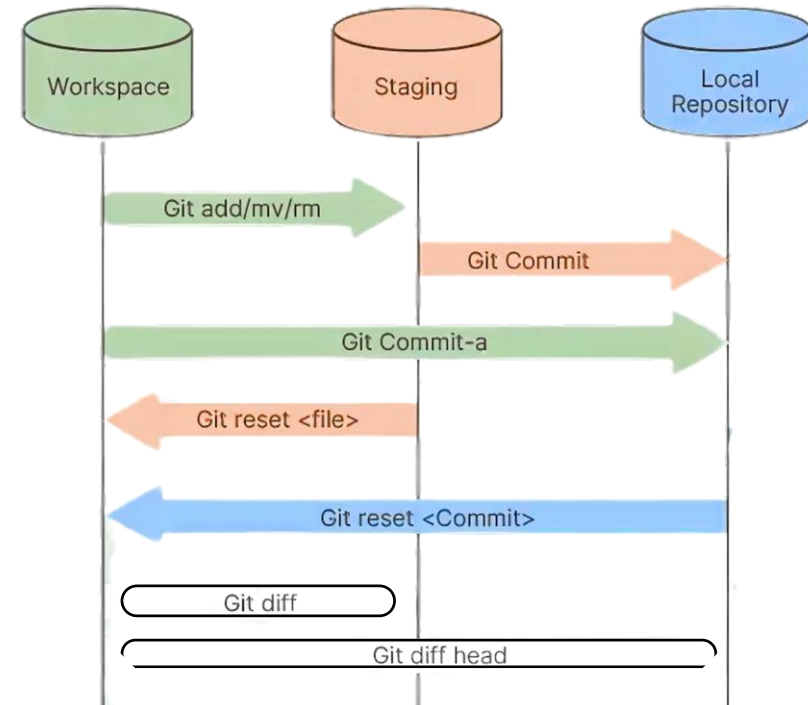
git mv filename1.txt filename2.txt (Note: File file1.txt was committed to repo earlier)

- Grabs the file from the repo, removing all changes since last commit

git checkout <filename>

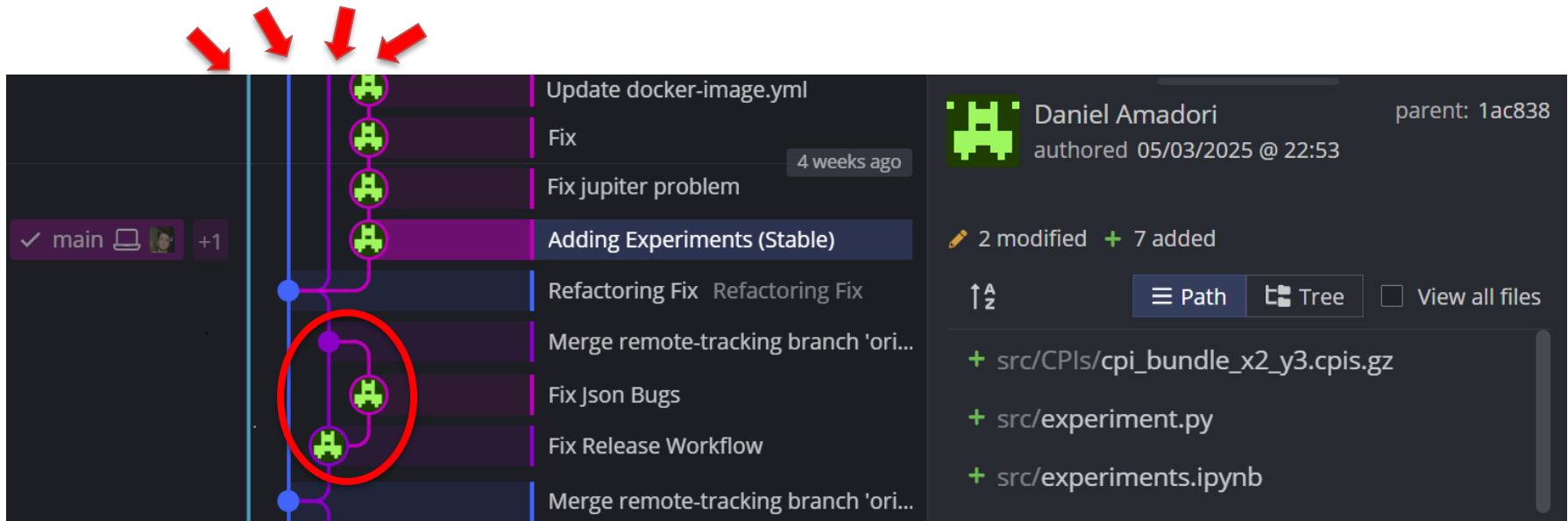
- Undo changes added to staging area

git reset HEAD <filename>



Branches in Git

- allows one to try new ideas
- If an idea doesn't work, throw away the branch. Don't have to undo many changes to master branch
- If it does work, merge ideas into master branch.



Branch Management

- Check current branch

`git branch`

- Create a new branch

`git checkout -b <branch>`

- Note: At this point, both HEADs of the branches are pointing to the same commit (that of master)

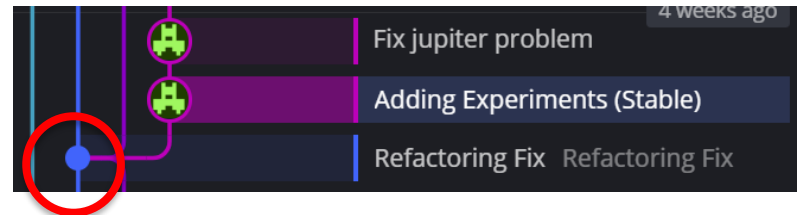
- Switch to another branch

`git checkout <branch>`

- At this point, one can switch between branches, making commits, etc. in either branch, while the two stay separate from one another.
 - Note: In order to switch to another branch, your current working directory must be clean (no conflicts, resulting in data loss).

- Compare two branches

`git diff <branch1>..<branch2>`

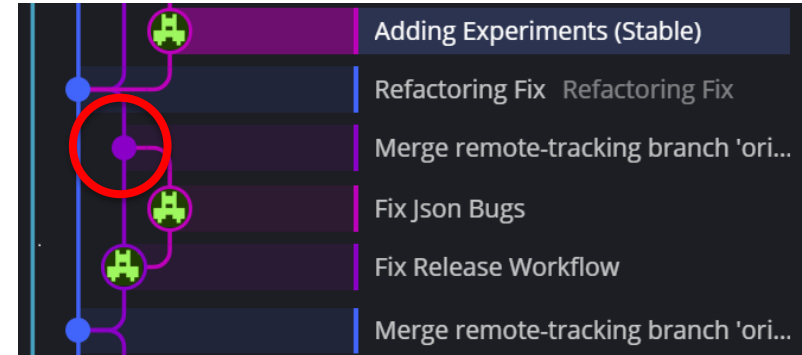


Merge Branches

- Merge from the branch into which you want to merge another branch

`git merge <branch to merge>`

- Note: Always have a clean working directory when merging

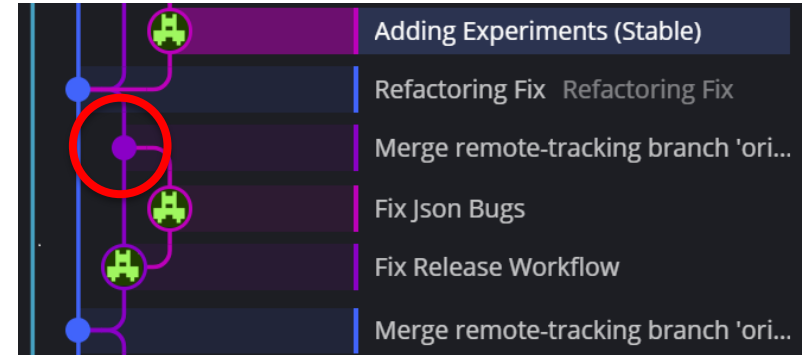


Merge Branches

- Merge from the branch into which you want to merge another branch

`git merge <branch to merge>`

- Note: Always have a clean working directory when merging



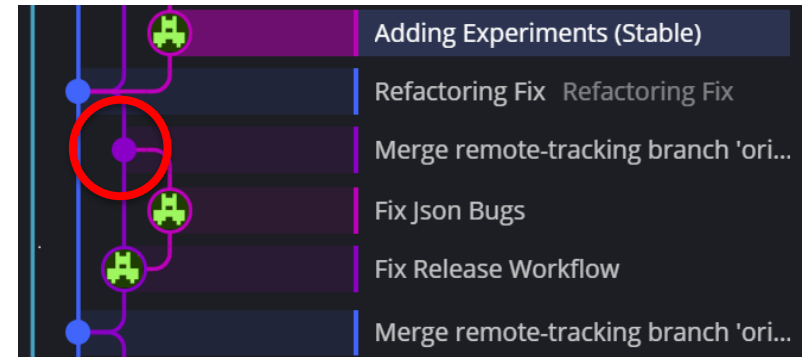
So easy?

Merge Branches

- Merge from the branch into which you want to merge another branch

`git merge <branch to merge>`

- Note: Always have a clean working directory when merging



So easy?

What if there are two changes to the same line in two different commits?

Merge conflicts



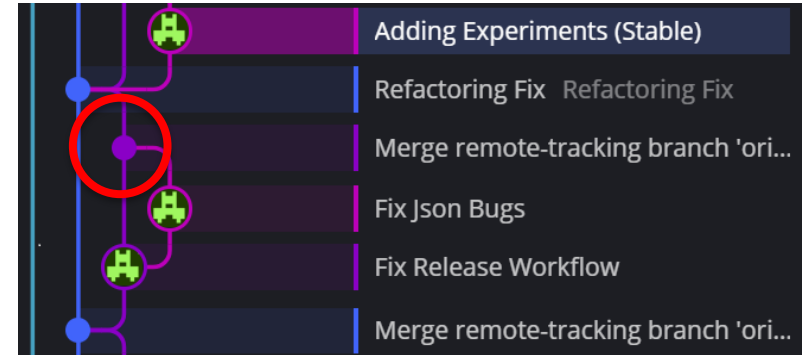
https://www.reddit.com/r/ProgrammerHumor/comments/cgf0b8/git_merge/#lightbox

Merge Branches

- Resolving merge conflicts

1. Abort the merge using
`git merge --abort`
2. Manually fix the conflict

Note: Git will notate the conflict in the files!



Understanding a Merge Conflict

- When Git detects conflicting changes, it marks them in your file using:
 - <<<<<< HEAD
 - Changes from your branch
 - =====
 - Changes from the other branch
 - >>>>>> branch-name
- You must manually resolve the conflict:
 - Choose one side's changes
 - Combine both
 - Rewrite the content
- Remove all conflict markers after editing.

Tips to reduce merge pain

- Merge Often

Integrate changes regularly to catch conflicts early, while they're easier to fix.

- Keep Commits Small & Focused

Write commits that solve one issue at a time, is easier to review, test, and roll back

- Regularly Pull from Main

Keep your feature branch up to date using 'git pull origin main' or 'git rebase' to reduce conflict risk.

Renaming and deleting branches

- `git branch -m/--move <old name> <new name>`
- `git branch -d <branch name>`

Note: Must not be in <branch name>

Must not have commits in <branch name> unmerged
in branch from which you are deleting

- `git branch -D <branch name>`

Note: If you are really sure that you want to delete branch with commits

Managing Work with Git Stash

- Allows you to save your work and change branch
git stash save "message"
 - Note: used for switching quickly between branches
don't leave the code in stash for a long period of time
- Lists all current stashes
git stash list
- Applies and deletes the most recent stash
git stash pop <index>
- Applies a stash but keeps it in the list
git stash apply <index>
- Creates a new branch starting from a stash
git stash branch <branch name> <index>
 - Note: useful for long-term feature development.

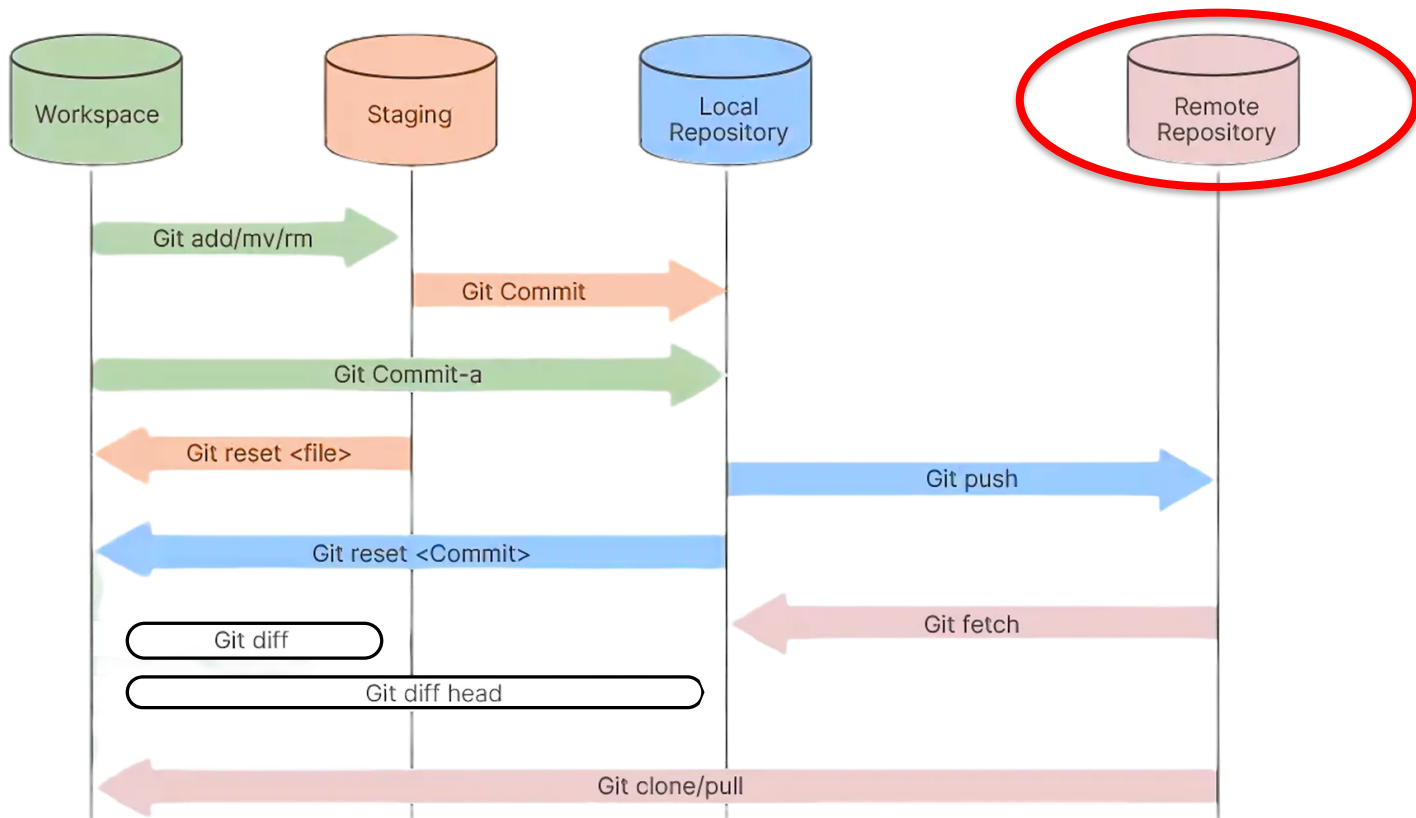
GitHub



<https://github.com>

- Platform to host code repositories
- launched in 2007
- most popular Git host
- allows users to collaborate on projects from anywhere
- GitHub makes git social!
- Free to start (can pay for private repositories and additional features)

Git architecture



Other Git Commands

- Copying (cloning) files from remote repo to local machine

`git clone URL <new dir name>`

- Pushing to a remote repo

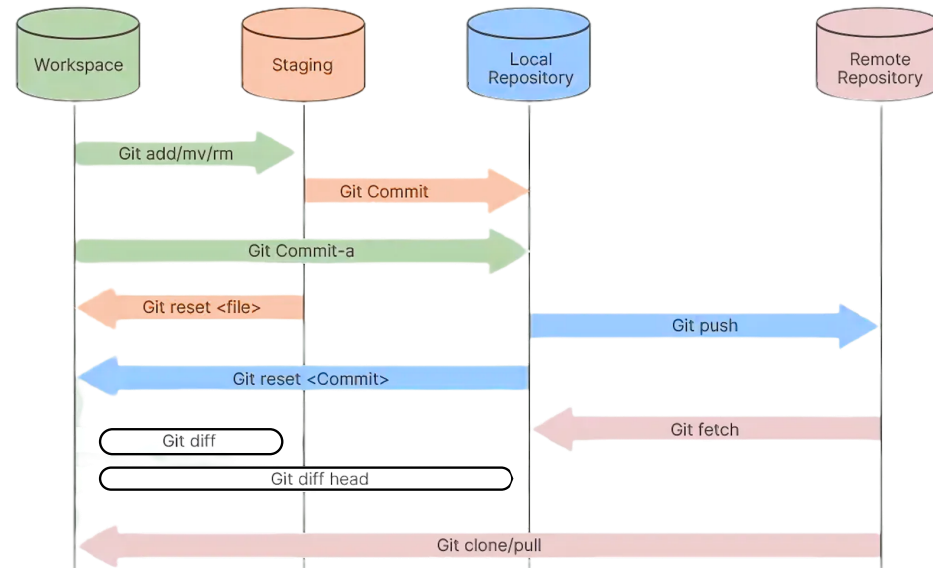
`git push <remote name> <branch name>`

- Fetch

`git fetch <remote repo name>`

Note:

- doesn't change your working dir or any commits that you've made
 - git merge must be done to merge fetched changes into local branch
- Fetch and merge changes to local branch and working directory
`git pull <remote repo name>`



Other Git Commands

- Link my local repo to a remote repo

`git remote add <alias> <URL>`

Note: just establishes a connection...no files are copied/moved

You may have more than one remote linked to your local directory!

- Tag specific points in history as being important, such as releases versions (v.1.0, 2.0, ...)

`git tag`

Forking a Repository (GitHub)

- Click “Fork” on a repository page on GitHub

- What it does:

Creates a personal copy of someone else's repository in your own GitHub account

Allows you to freely experiment, edit, or contribute without affecting the original

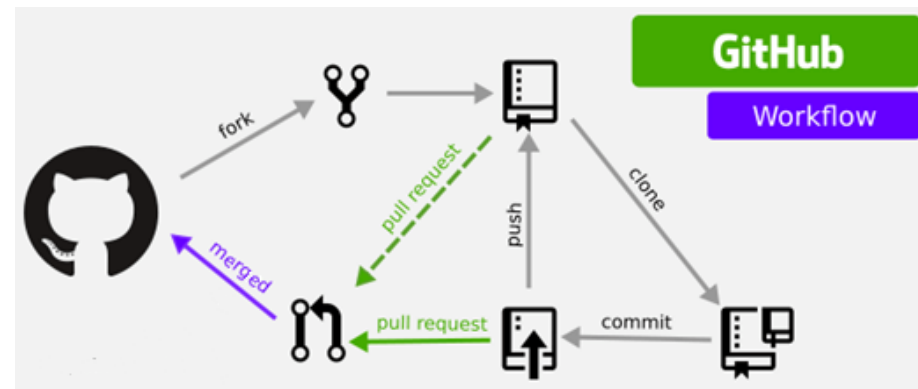
- Why it matters:

Essential for open-source collaboration

Used in the 'Fork and Pull Request' workflow

- Typical flow:

1. Fork a repo on GitHub
2. Clone your fork locally:
3. Create a branch, make changes, push, then open a pull request



<https://levelup.gitconnected.com/how-to-sync-forked-repositories-using-git-or-github-2933e497fa16>

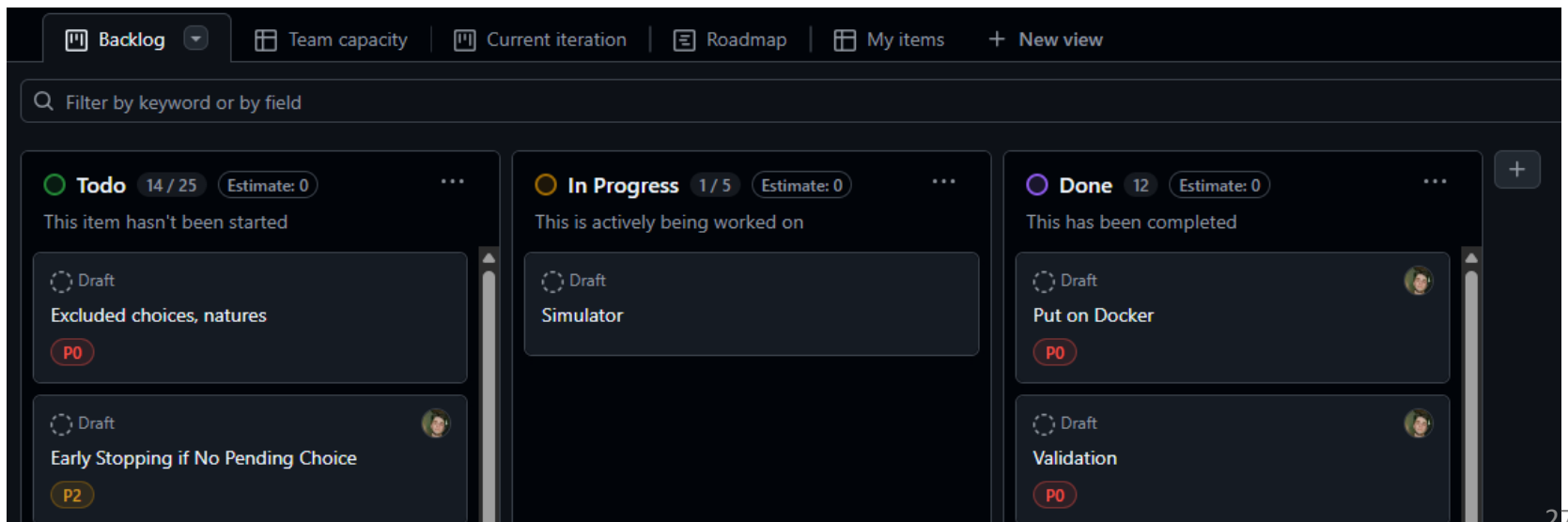
GitHub Projects (Backlog)

Backlog:

- prioritized list of work items that need to be completed (features, bugs, and tasks)

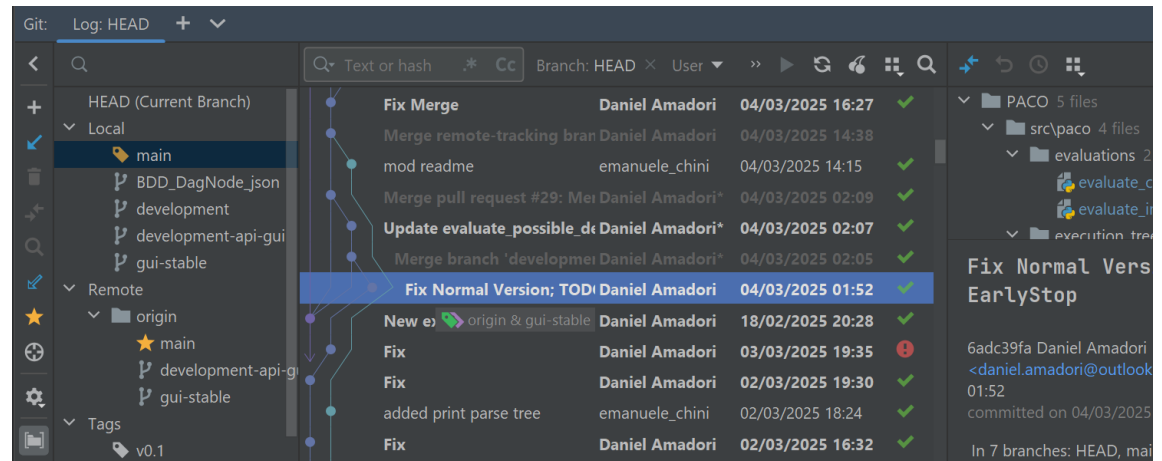
In Agile:

- Contains user stories, tasks, and issues
- Continuously updated and reprioritized

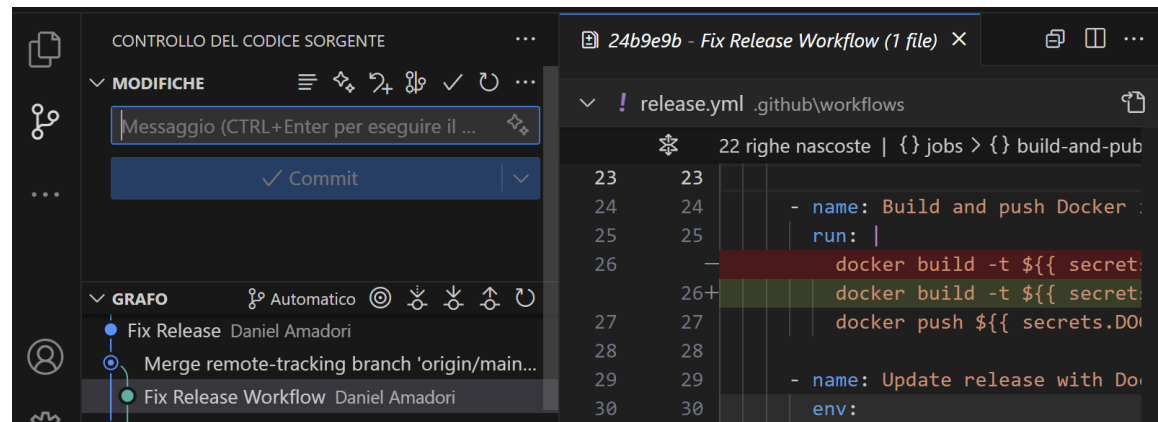


Git in IDE

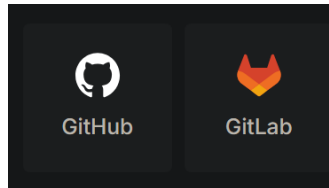
PyCharm



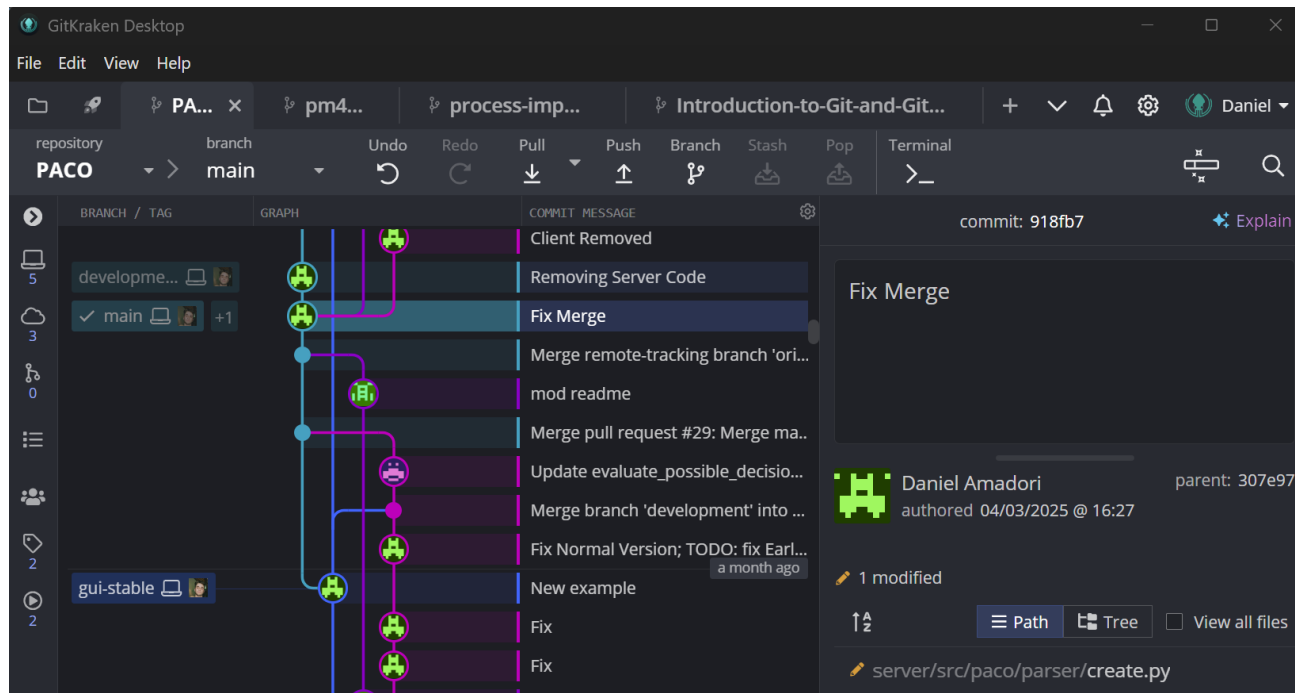
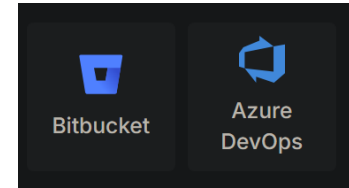
Visual Studio



Repo Management



GitKraken



Resources

- Git: <https://git-scm.com/>
- Try GitHub: <https://try.github.io/>