GitHub Basics Workshop

Sinclair Combs

July 23rd, 2025

Git is about keeping track of changes

Version Control

My code broke… hopefully there's a working version somewhere

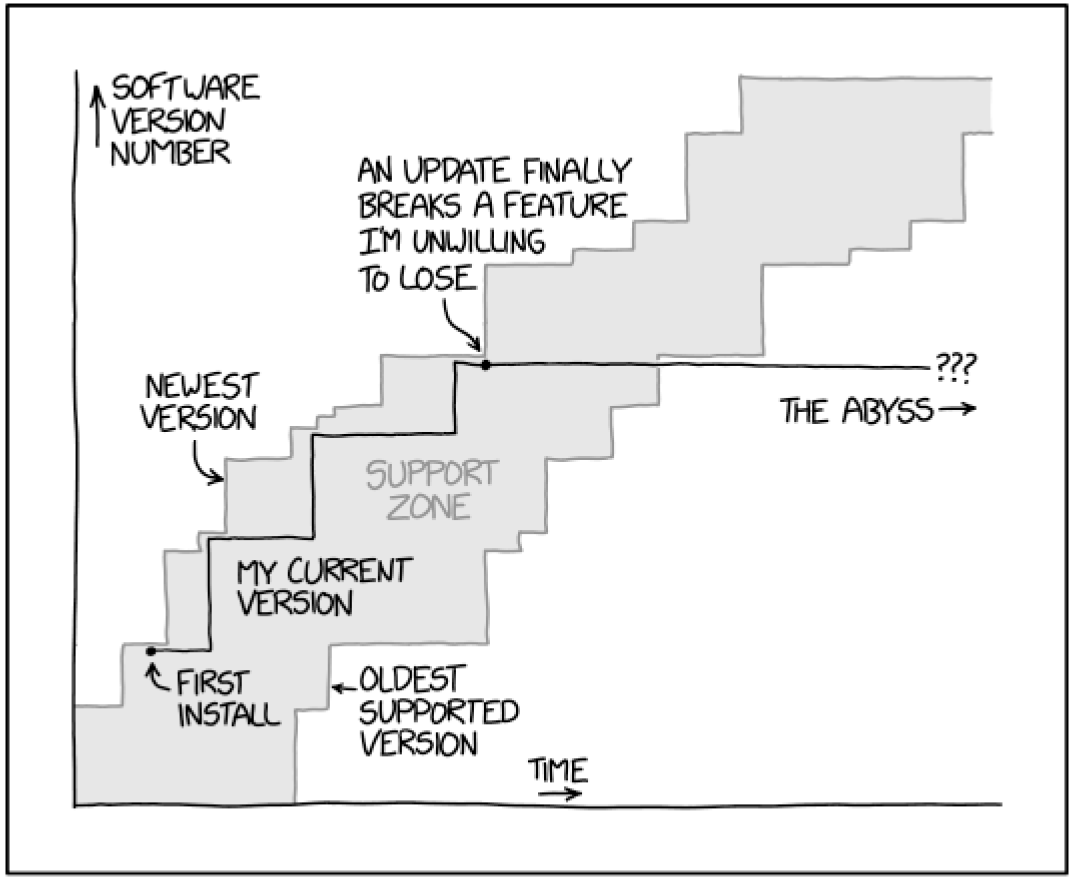
Which version are you using?

Can I reproduce my results?

I'm sure this used to work… when did it change?

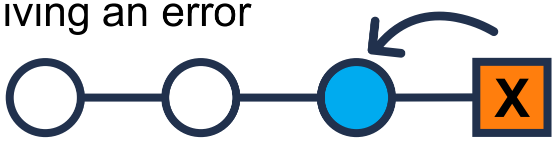
How long has this bug been here?

Can you send me the latest version?



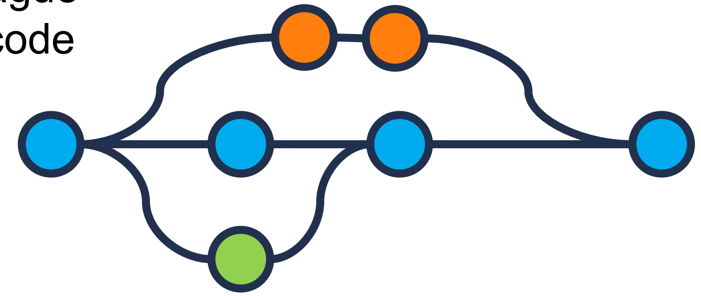
ALL SOFTWARE IS SOFTWARE AS A SERVICE.

Problem: Code worked two days ago, but is now is giving an error



Solution: roll-back; return to previous version & compare

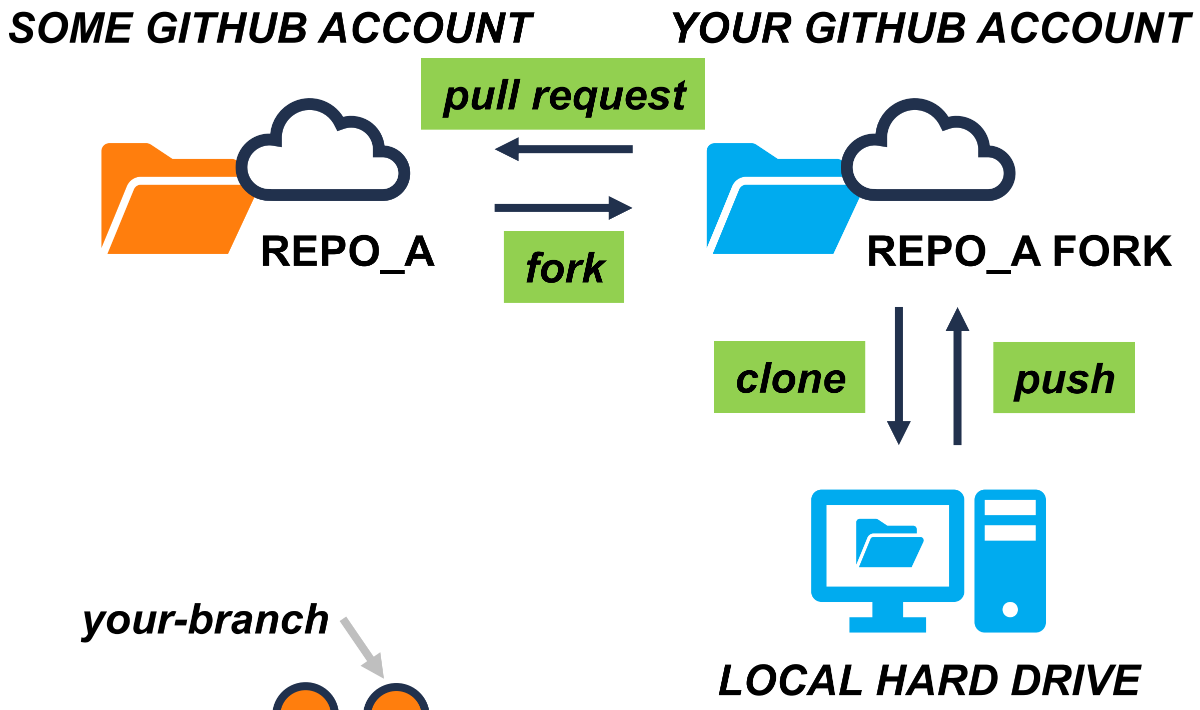
Problem: You & your colleague want to work on the same code at the same time

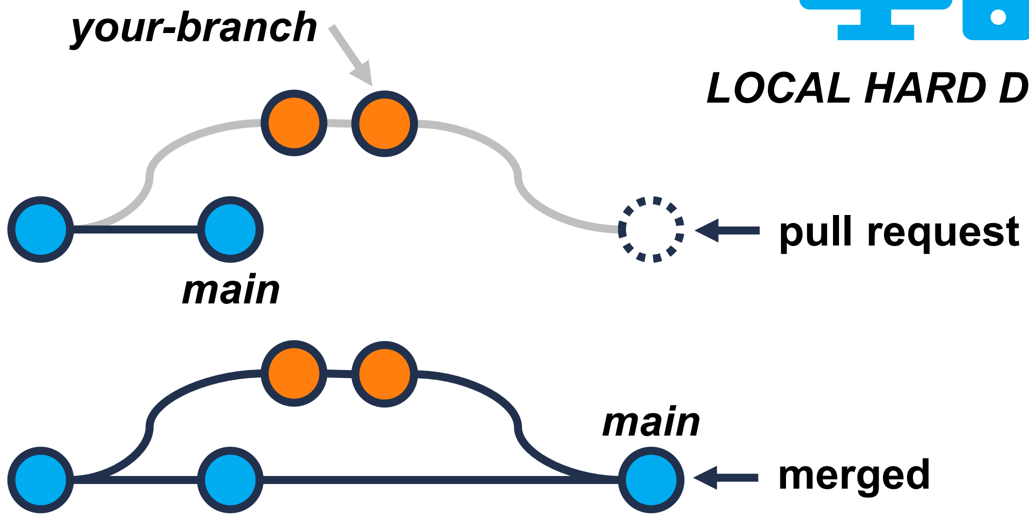


Solution: branching & merging; simultaneous work without interference

Some version control vocabulary

Repository	Collection of files and their history
Fork	A specific copy of a repository that is linked to the original, making it simple to send changes upstream
Clone	Local copy of a repository, including history
Push	Moving changes from a local copy to another copy
Pull	Getting changes from another copy to a local copy
Pull Request	Change proposal to merge one branch into another, usually upstream



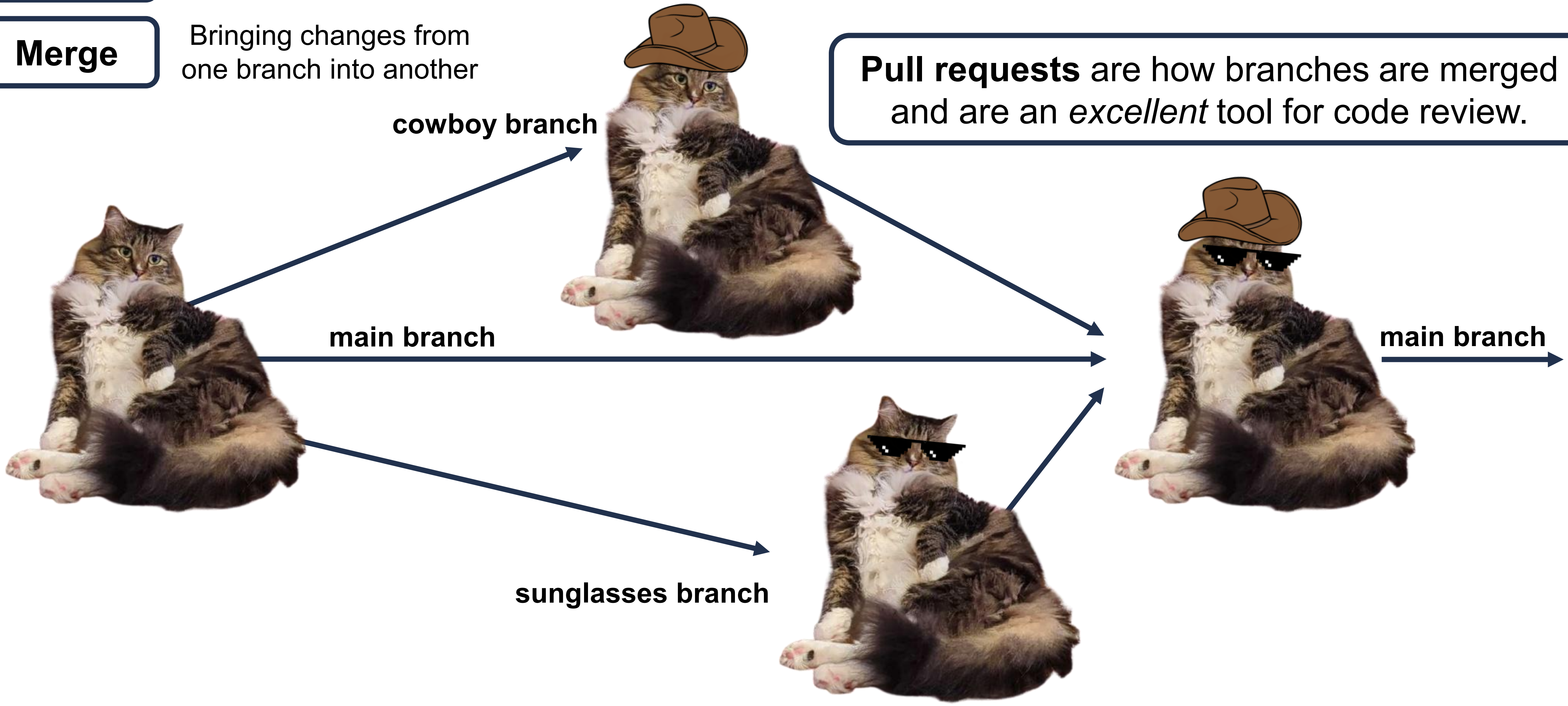


Basics of branching & merging

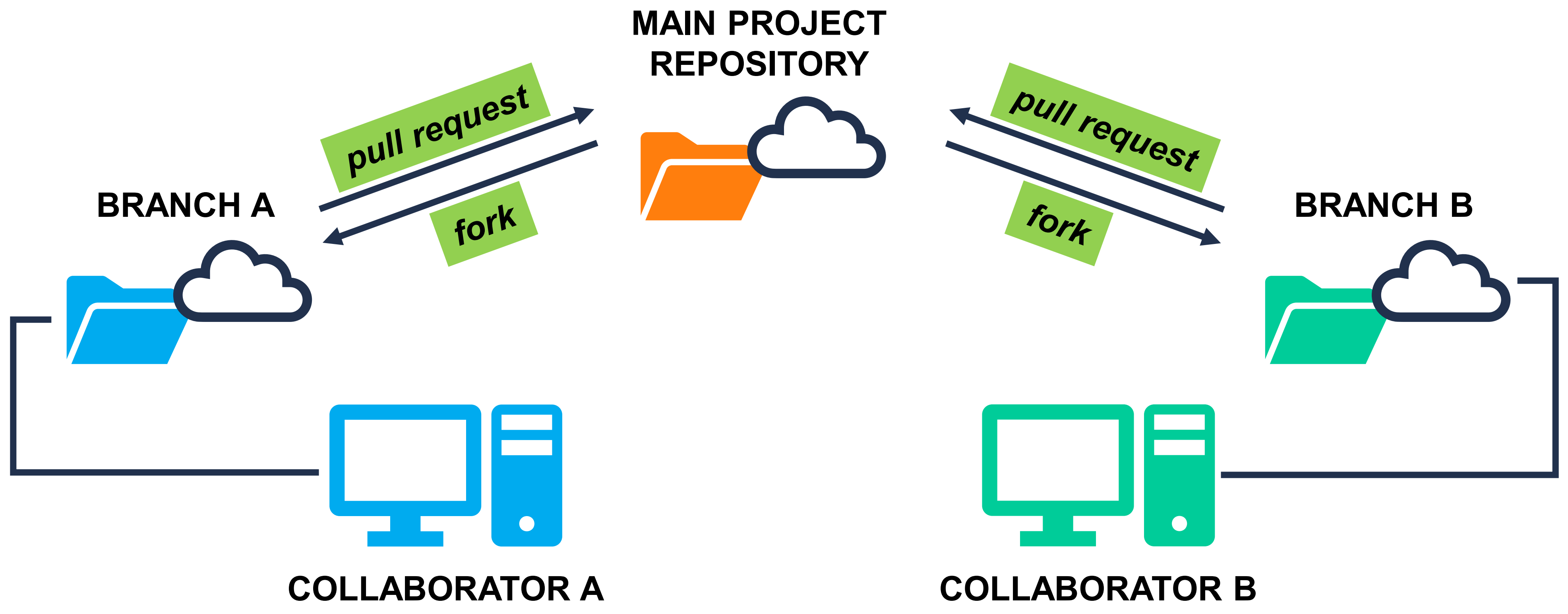
Branch One line of work; different branches can exist at the same time and split/merge

Merge Bringing changes from one branch into another

Pull requests are how branches are merged and are an *excellent* tool for code review.

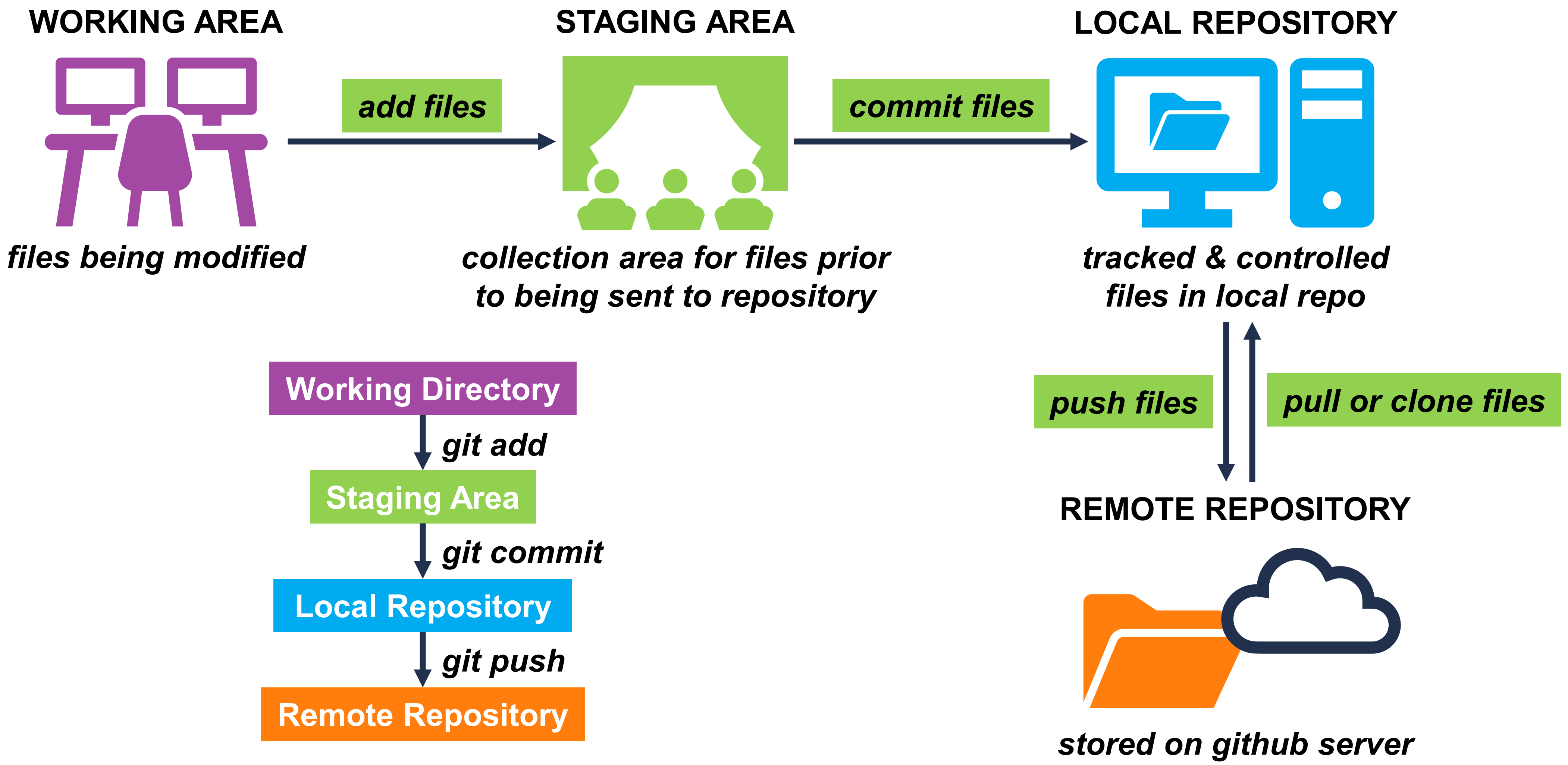


Forking workflow for collaboration

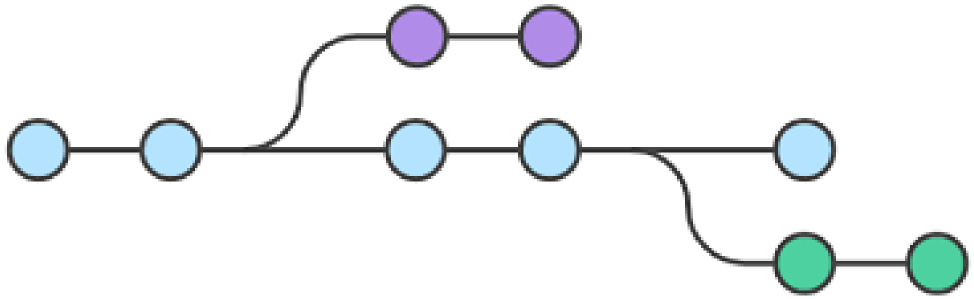


- Everyone has a fork of a “central” repository
- Add commits to branches for new features
- Send pull requests from feature branches to the central repository

Recording changes with commit

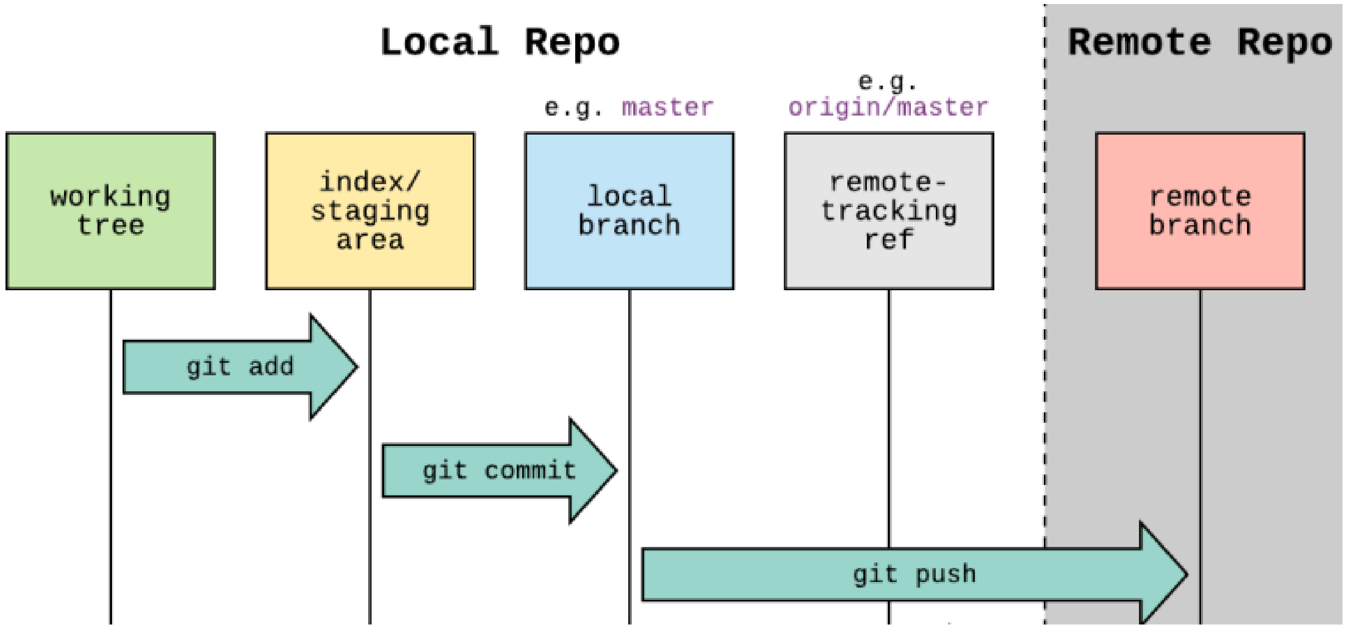


How do I use branches and commits effectively?



- Use branches for **unfinished** or **untested** ideas
- Use branches when you are **not sure about a change**
- If you are unsure what to do with unfinished and/or not working code, commit to a branch
- Use branches for code that will be **reviewed** by others

- Commit **early** and **often**, it's better to create too many commits than too few
- Better too **small** than too large
- Once you commit, it's very hard to lose code
- Always fully commit before doing dangerous things
- Good practice to commit after ending any session of coding
- **Imperfect commits are better than no commits**
- Do not commit unrelated changes together, this makes it difficult to undo changes

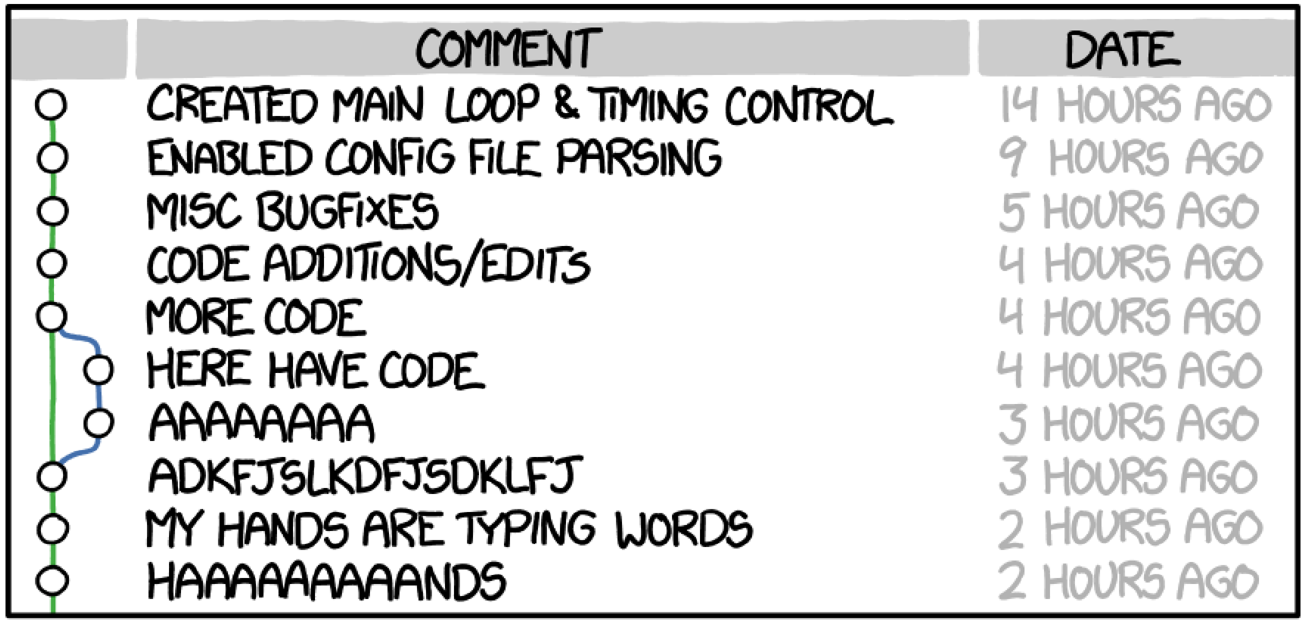


Writing useful commit messages

Summarize the change & provide context

- **Why** something changed is usually more important than **what** has changed
- Cross-reference to issues/discussions if relevant
- Write messages that will be understood 20 years from now by someone other than you. Or your future self.
- Bad commit messages: “fix”, “oops”, “save work”

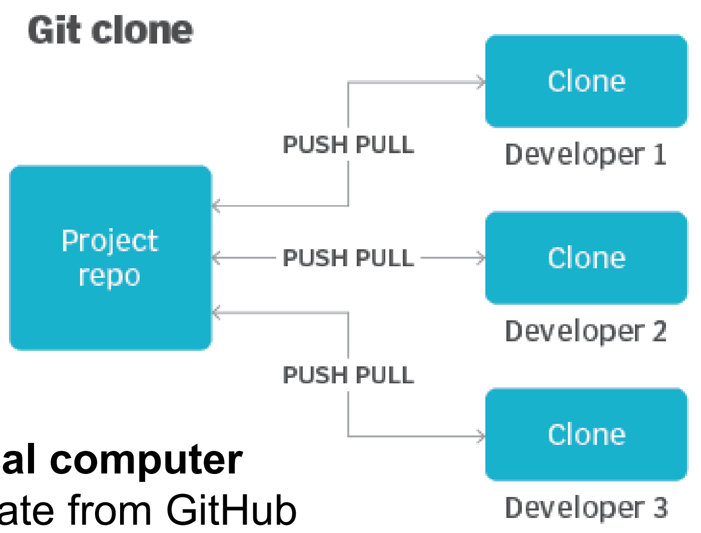
whatthecommit.com



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

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

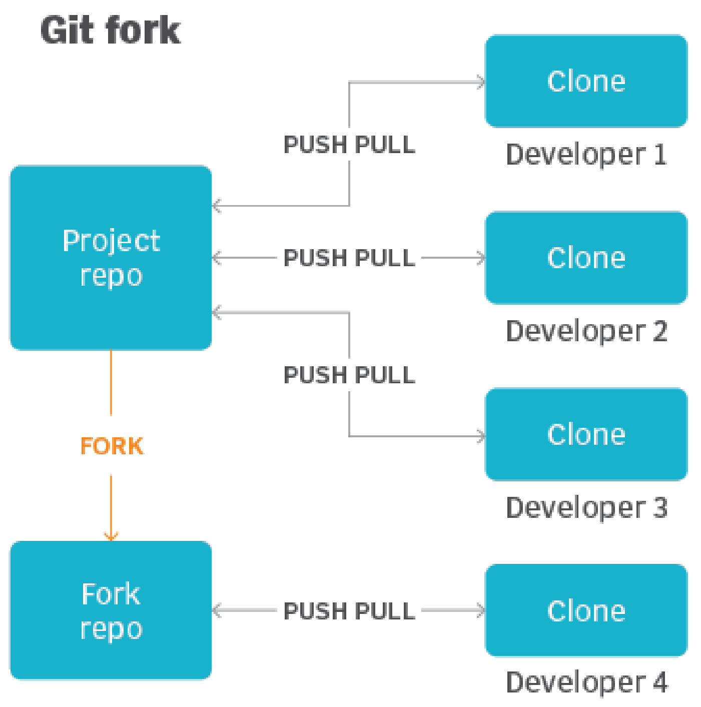
Collaborative workflow—Cloning vs. forking



- Copied to a **local computer** and lives separate from GitHub
- Relevant for small team development where everyone has **read and write access** to the parent repository

Remember: Forked or cloned repositories do not automatically synchronize themselves!

- We **pull** updates **from** remote repositories
- We **push** updates **to** remote repositories
- We can **suggest changes within** and **across** repositories on GitHub with **pull requests**



- Copied to a **different <u>GitHub</u> account**
- Relevant for working on a repository to which you **do not have write access**
- In the fork, commits can be made to the main branch
- Commits from the fork can be contributed back to the parent repository via pull requests

Tutorials!

Tutorial topics:
1. Copy and browse
2. Commits
3. Merging
4. Sharing

github.com/Maughan-Lab/github_workshop/tree/main/tutorials