Quick intro to Git

A quick intro to Git

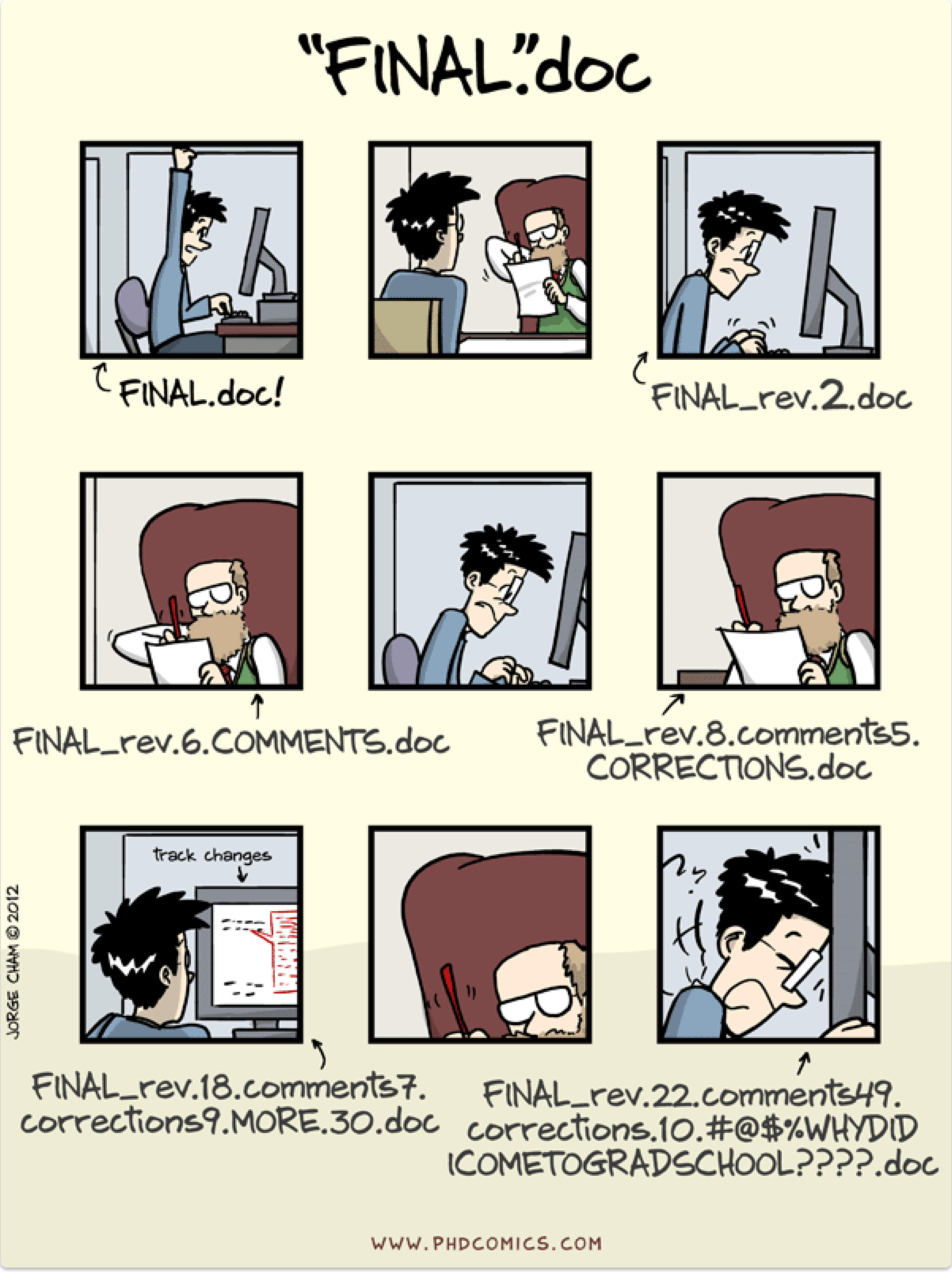
Pin Shuai [1]

November 3, 2023

What is Git?

- Version control tool that tracks file change history
- Popular platform includes GitHub, GitLab, BitBucket ...

Why should I use Git?

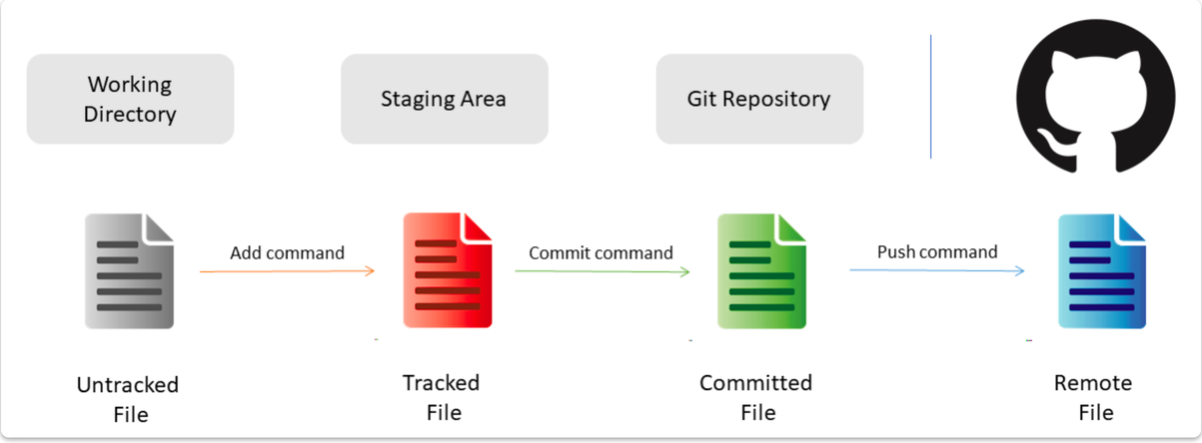


Basics of Git

Three Stages

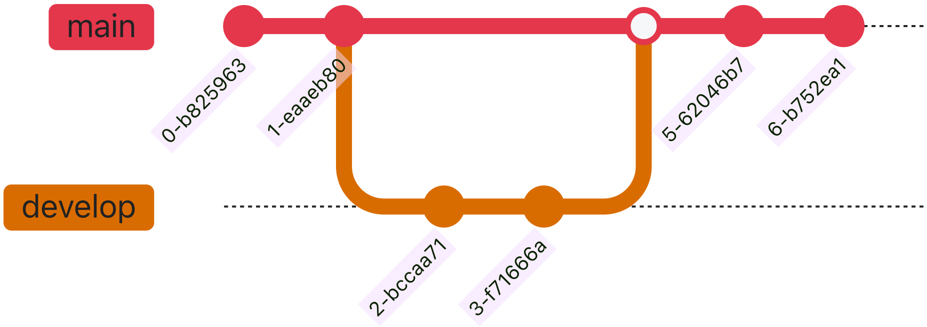
- Untracked/modified: the file is new or modified, but is not part of git's version control

- Staged: the file has been added to git's version control but changes have not been committed
- Committed: the change has been committed (created a new version)



Branches

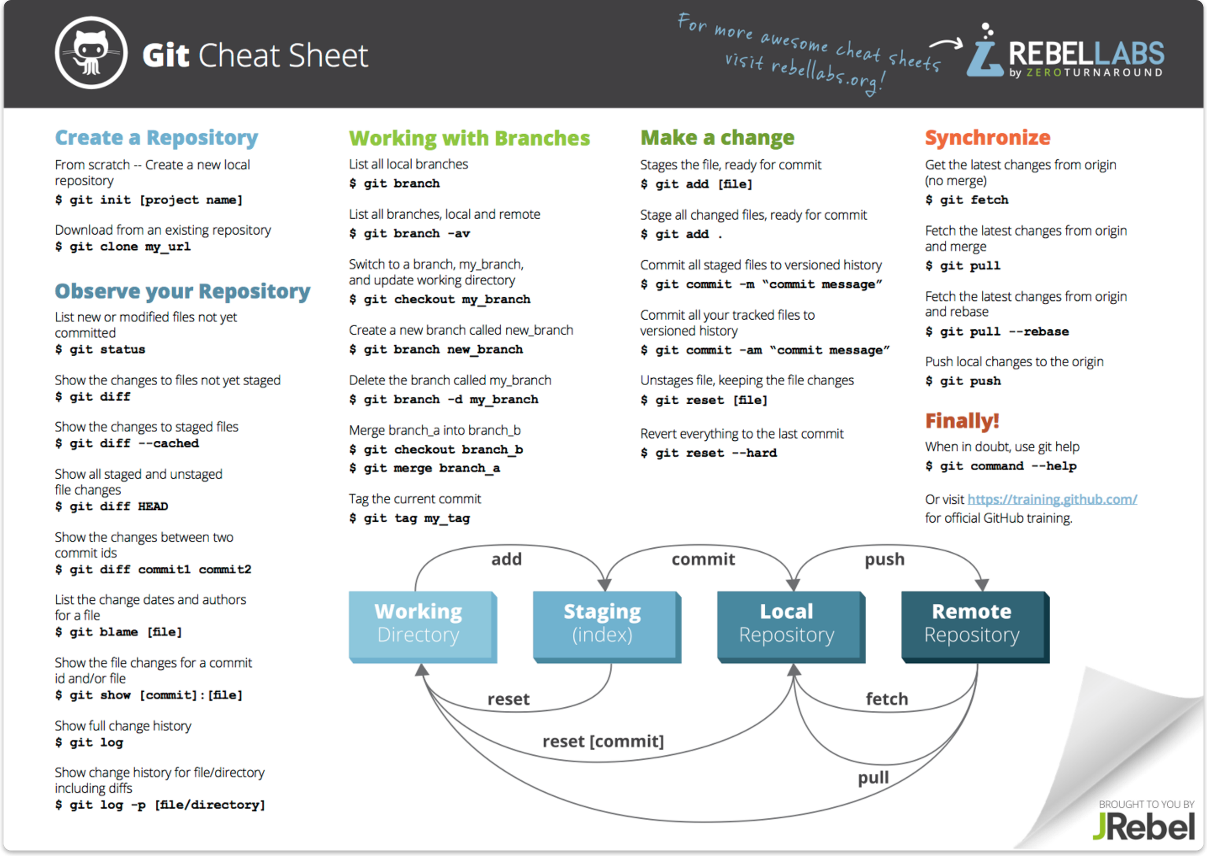
- Master (aka Main): the default branch
- Develop: adding new features or fix bugs



Common commands

- `git status`: show file status (untracked, modified, comitted, etc.)
- `git add FILENAME`: add untracked files to staging area
- `git commit -m YOUR_MESSAGE`: commit the staged files to local repository (or version history)
- `git pull`: fetch latest changes from remote repository (always pull before push)
- `git push`: push local changes to remote repository (e.g., GitHub)

Cheatsheet



Git Cheat Sheet

For more awesome cheat sheets visit rebellabs.org!

REBELLABS by ZEROTURNAROUND

Create a Repository

From scratch -- Create a new local repository
`$ git init [project name]`

Download from an existing repository
`$ git clone my_url`

Observe your Repository

List new or modified files not yet committed
`$ git status`

Show the changes to files not yet staged
`$ git diff`

Show the changes to staged files
`$ git diff --cached`

Show all staged and unstaged file changes
`$ git diff HEAD`

Show the changes between two commit ids
`$ git diff commit1 commit2`

List the change dates and authors for a file
`$ git blame [file]`

Show the file changes for a commit id and/or file
`$ git show [commit]:[file]`

Show full change history
`$ git log`

Show change history for file/directory including diffs
`$ git log -p [file/directory]`

Working with Branches

List all local branches
`$ git branch`

List all branches, local and remote
`$ git branch -av`

Switch to a branch, my_branch, and update working directory
`$ git checkout my_branch`

Create a new branch called new_branch
`$ git branch new_branch`

Delete the branch called my_branch
`$ git branch -d my_branch`

Merge branch_a into branch_b
`$ git checkout branch_b`
`$ git merge branch_a`

Tag the current commit
`$ git tag my_tag`

Make a change

Stages the file, ready for commit
`$ git add [file]`

Stage all changed files, ready for commit
`$ git add .`

Commit all staged files to versioned history
`$ git commit -m "commit message"`

Commit all your tracked files to versioned history
`$ git commit -am "commit message"`

Unstages file, keeping the file changes
`$ git reset [file]`

Revert everything to the last commit
`$ git reset --hard`

Synchronize

Get the latest changes from origin (no merge)
`$ git fetch`

Fetch the latest changes from origin and merge
`$ git pull`

Fetch the latest changes from origin and rebase
`$ git pull --rebase`

Push local changes to the origin
`$ git push`

Finally!

When in doubt, use git help
`$ git command --help`

Or visit https://training.github.com/ for official GitHub training.

BROUGHT TO YOU BY JRebel

[Git Cheat Sheet : r/git](#)

Collaboration through Git

- A typical workflow looks like this:
  - Create your GitHub account at https://github.com
  - Fork a git repository
  - Open the repository on GitHub or locally through GUI (e.g, [GitHub Desktop](#)) or terminal (e.g., PowerShell)
  - Create a new branch (e.g., `git checkout -b NEW_BRANCH`)
  - Make some changes, commit and push
  - Submit a PR (pull request)
  - After review, a PR is approved by the original repository owner

---

Demo

Homework

- Fork the group website repository on GitHub: https://github.com/hydroaggie/hydroaggie.github.io

- Add information to your personal page via a PR

References

- Software Carpentry: Version Control with Git: Summary and Setup
- Git - Documentation

[1] Utah State University↩