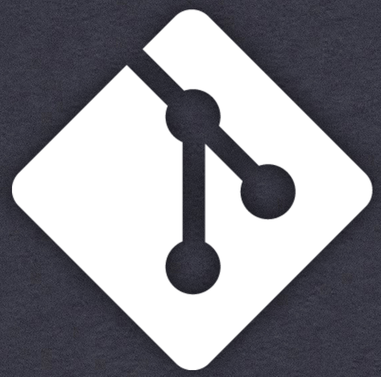


git GitHub

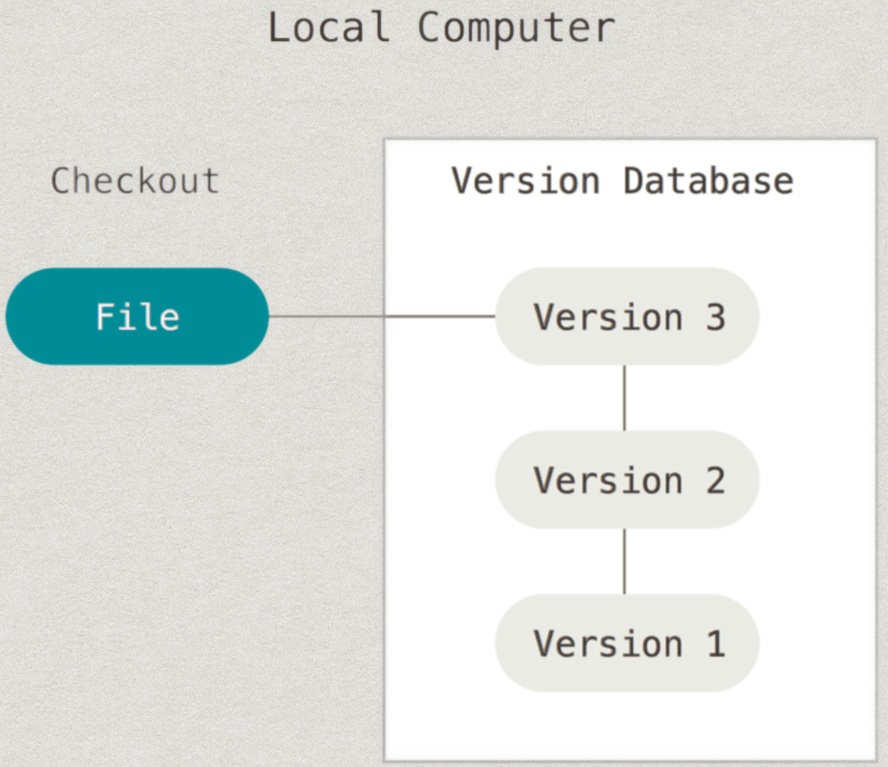
GIT & GITHUB

WHAT ARE THEY, WHEN TO USE THEM, WHY WE USE THEM, HOW TO MASTER THEM

WHAT IS GIT?

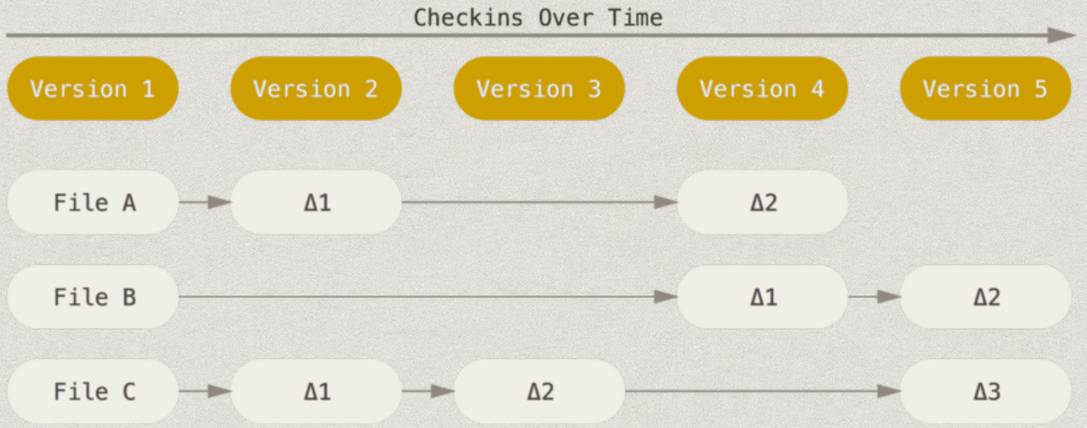
Git

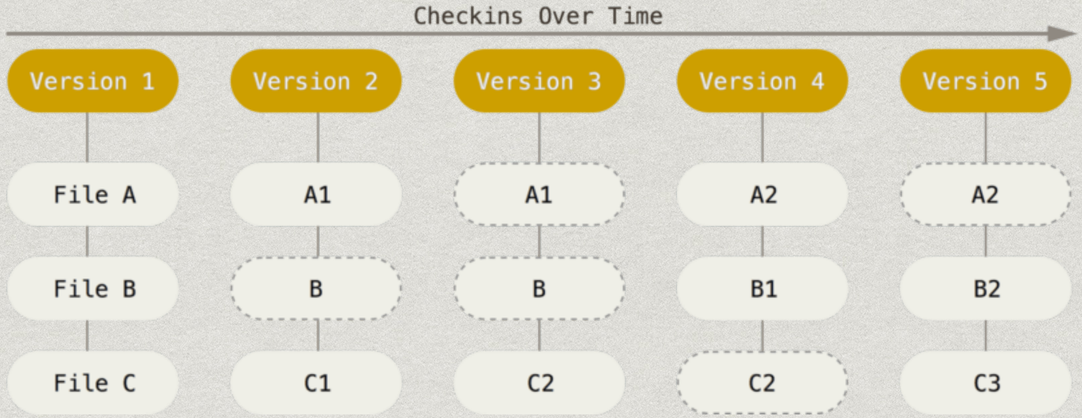
- Git is a VCS (Version Control Software)
- Is small and portable
- Free (as in freedom) and open source
- Comes standard with many common operating systems and their distributions



Why Git?

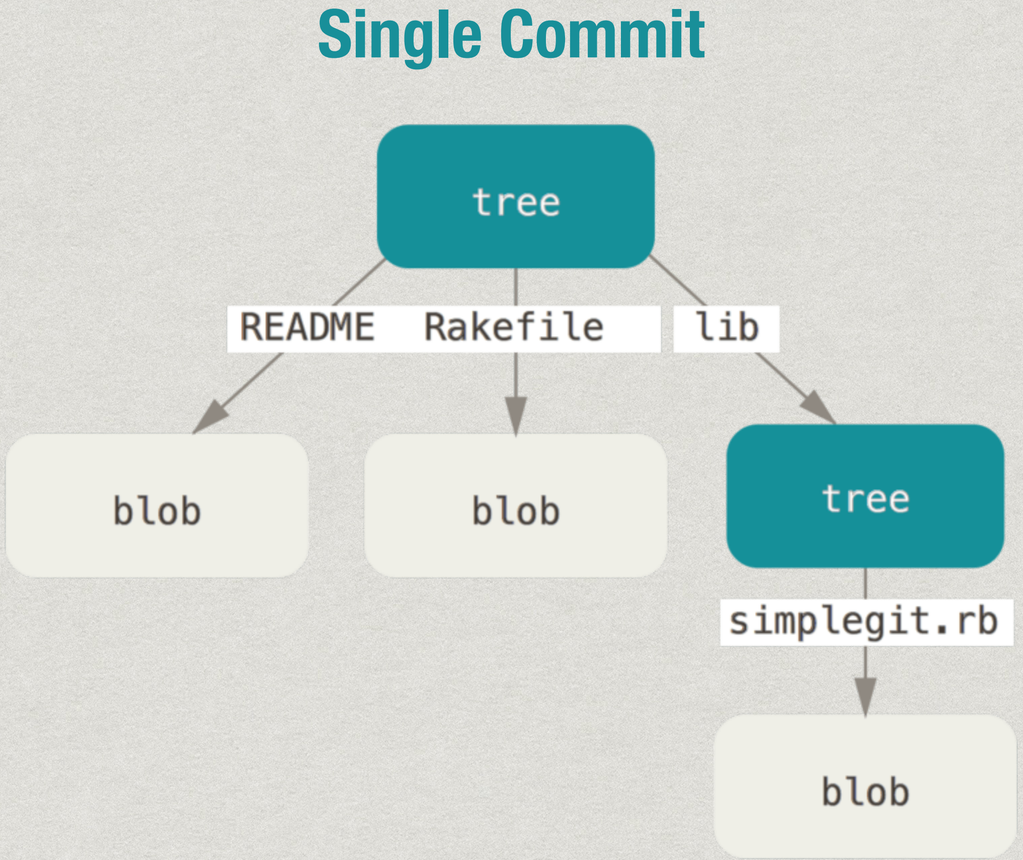
- Git is one of the most popular VCS software common across many industries.
- Easy to use, install and compatible with many other softwares like IDEs and text editors.
- Reliable, fast and secure.
- FOSS!





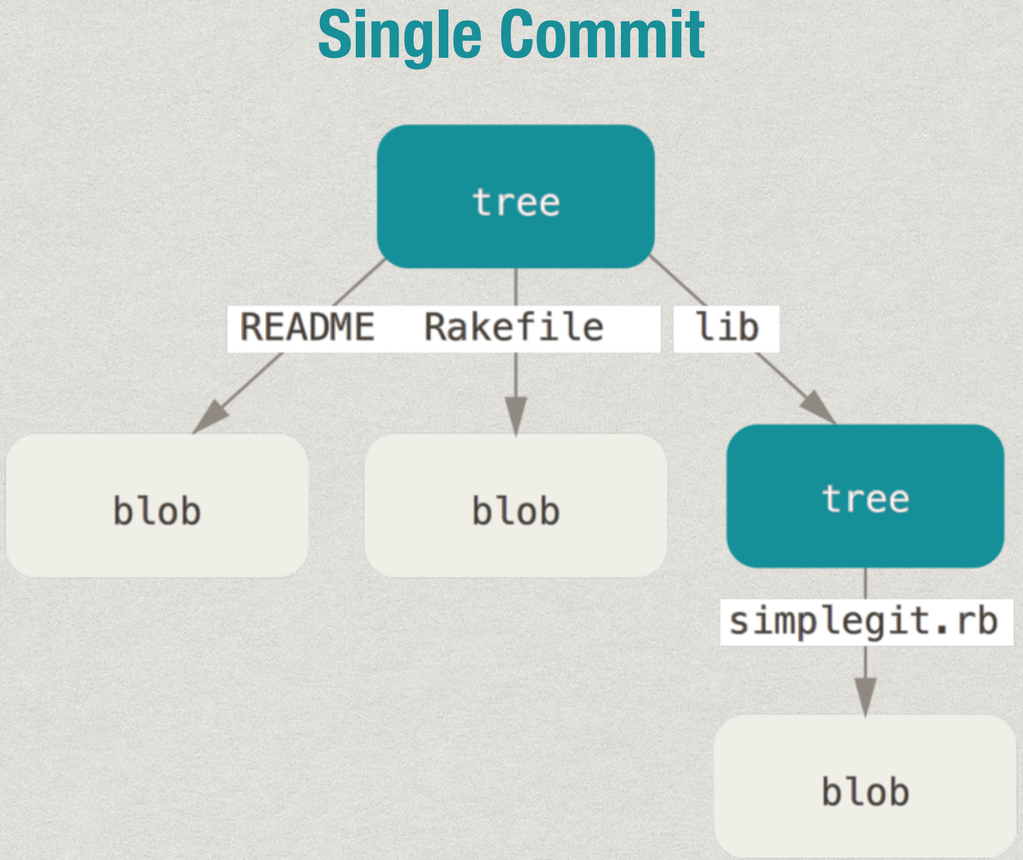
How does it work?

- Git has 4 types of objects to construct it’s version trees and work history these are:
  - Trees, Blobs, Tags, Commits
- Git uses Trees and Blobs to create a Commit object using a hashing algorithm to allocate a compressed version of the files changed into a folder under “.git/objects/“.
- The most important thing to take into account is that git manages project versions using Commits which are in turn composed of Trees and Blobs.



How does it work?

- Blobs are copies or versions of individual files that have been included in a commit.
- Trees are created by hashing a collection of blobs together and bundling them into a single SHA-1 key. They are akin to a UNIX filesystem in the sense that they are in charge of storing “folders” or rather the correct term "UNIX directories”.
- Commits are akin to a snapshot of the current state of the directory.



WHEN & WHY SHOULD I USE GIT?

- Creating multiple files for different versions or revisions of a file or folder.
- Having the need to ‘travel back in time’ to a previous version of a file or project.
- Keeping track of progress and work done.

If you’ve had any of these needs in a past project, its time to use a VCS (like git)

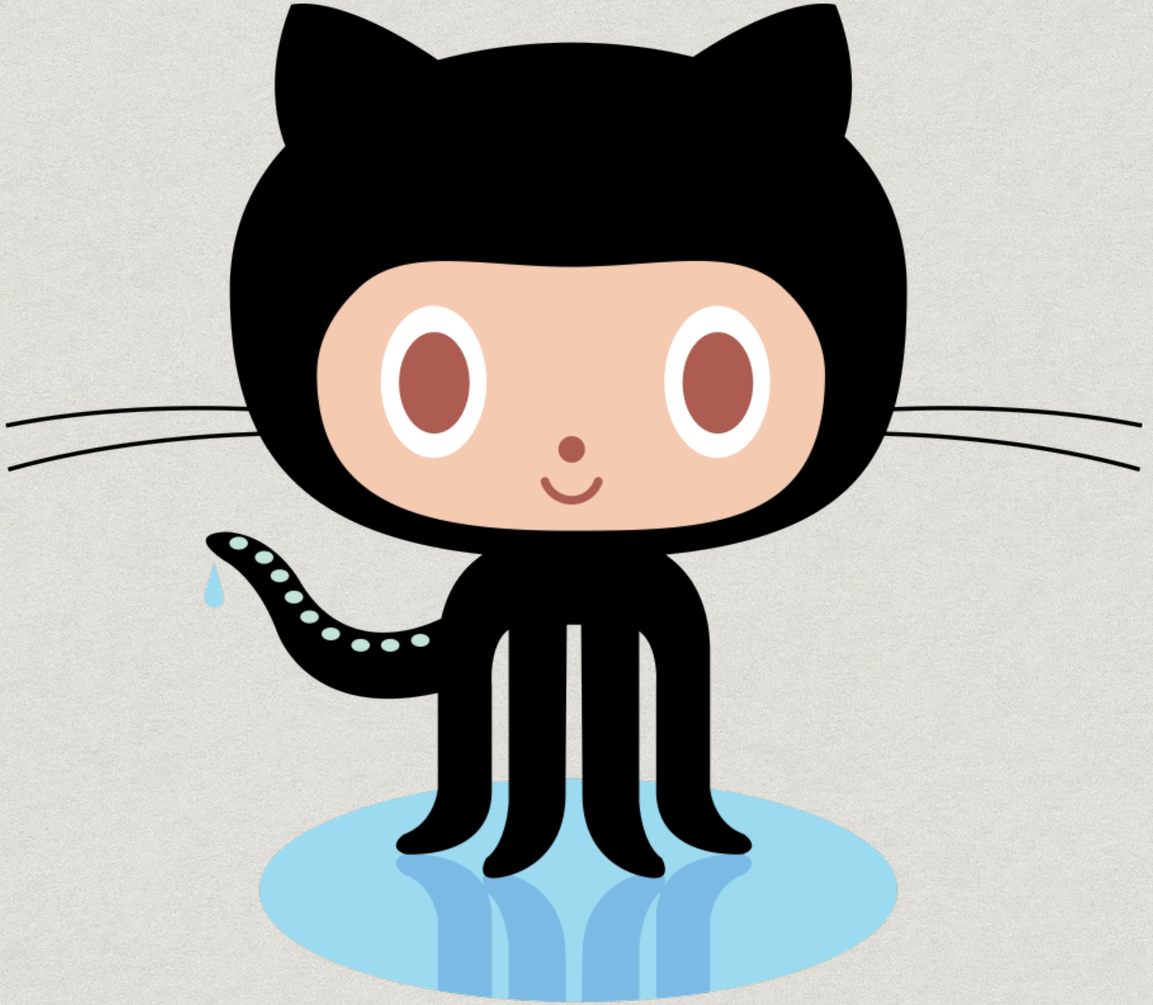
Other than Git?

- Apache Subversion
- Concurrent Version System
- Mercurial
- Monotone
- Perforce
- Fossil
- Bazaar

WHAT IS GITHUB?

Github

- Github is akin to a ‘social-network’ of software and technology.
- Github allows for easy collaboration between people and teams alike.
- Github is a Internet Hosting Platform readily available to host your git projects hence the name ‘git-hub’.
- A place where git repositories are hosted and shared on the internet.



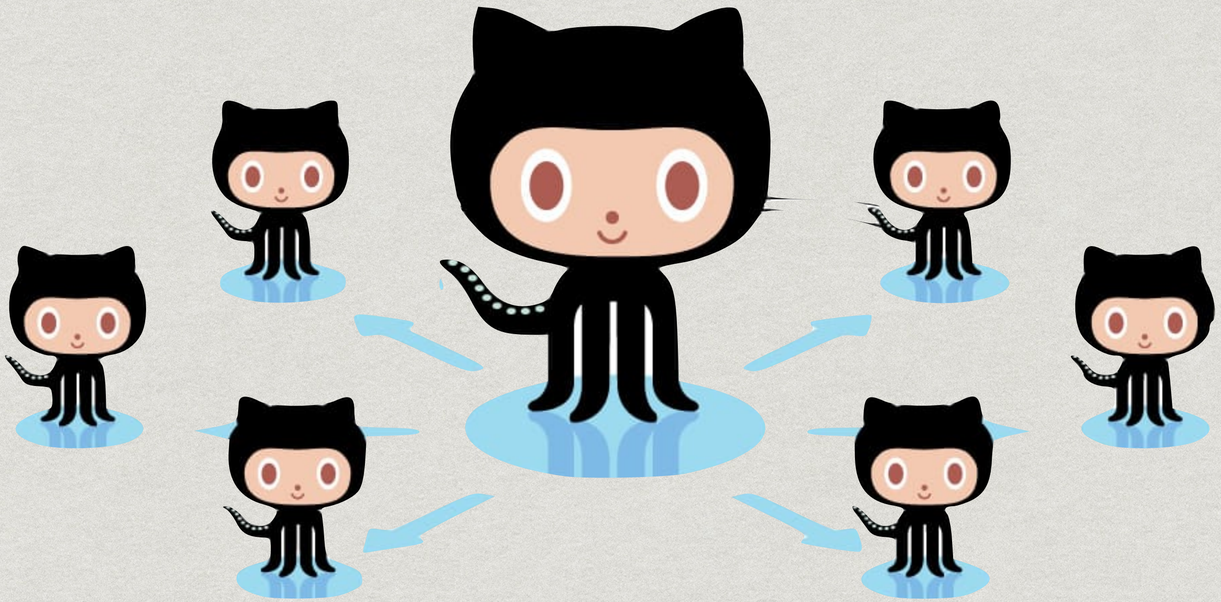
https://octodex.github.com/

Common Misconceptions

- Git is a completely different piece of software independent of Github.
- They have completely different development teams and managed by different companies.
- Git is Free (as in freedom) and Open Source released under the GPL v2 license, while Github itself is closed source and freemium (as in use and price)

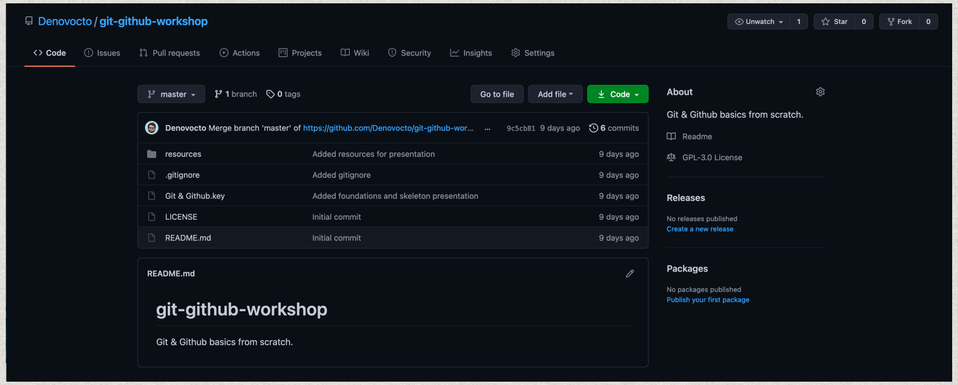
What can it do?

- Workflows (CI/CD)
- Project Management
- Pull Requests
- Issues
- Static Webpage Hosting
- Webhook integrations
- Environments



How to use it?

- Github is an online platform, thus you need to create an account with them in order to create or edit repositories on the cloud.
- One can create an account by going to https://github.com and signing up.
- Right after you have created an account you can interact with other people's and organization's repositories by giving repositories a star, forking them, watching them, commenting and/or creating issues.
- As well as interacting with pre-existing repositories you can create your own and upload your code into the cloud.



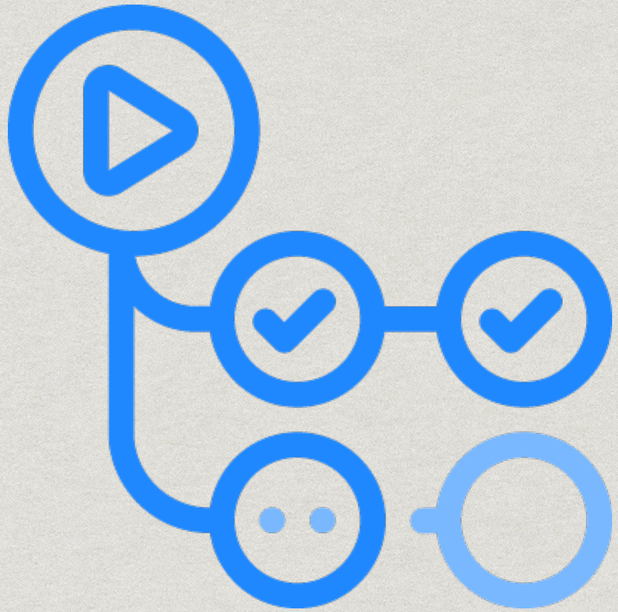
WHEN & WHY SHOULD I USE GITHUB?

Reasons to use Github

- Explore and discover projects and software that either you already use or find interesting to learn more about.
- Collaborate with open source projects or create your own!
- Share with other creators and collaborators.
- Use your profile and repositories to host your own projects and workflows.
- Use your profile as a portfolio for interviews and job openings!
- Github is very popular and used worldwide by a plethora of companies.

When to use GitHub

- When you have a project that requires collaboration between team members or teams.
- When you want to release your open source project for the whole world to see.
- When you want to benefit your project of any of the services that GitHub offers.



All of these suggestions are subject to your project's needs & license

Other than Github?

- Gitlab
- Bitbucket
- SourceForge
- Self-hosted repositories such as those of gcc or freeBSD

GitLab

GIT'IN GIT

Where do I start?

- If you are on windows you must first install git.
  - Recommendations for Windows users are either to use:
    - Install the Chocolatey package manager for windows and then install git with Chocolatey (It's managed by Microsoft)
      - In powershell: `choco install git.install`
    - Install git directly from the git webpage.

Where do I start?

- If you are on a OS X or Linux/UNIX-like/POSIX system, you should first check if it is already installed by heading to a terminal program and typing the following command:
  - `git --version`
  - If you see output from the command stating the version of git installed then you have it installed.
  - You should see something similar to (The version and distribution may change):
    - `git version 2.24.3 (Apple Git-128)`

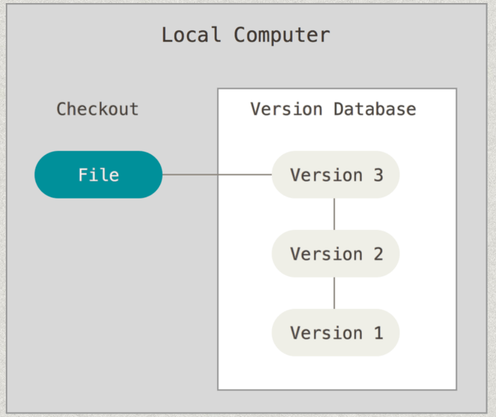
Where do I start?

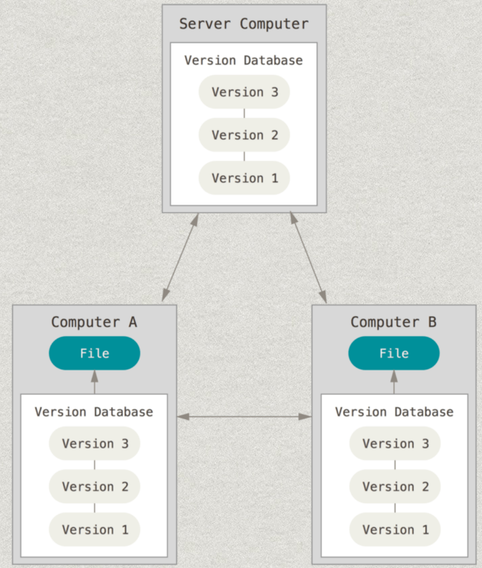
- If you see an error telling you that the command doesn’t exist similar to:
  - `command not found: git`
  - You need to install git first, to do so, refer to the package manager of your operating system in order to install.
  - Some examples of common package managers are:
    - sudo apt-get install git
    - brew install git
    - sudo pacman -Syu install git

GIT’IN' GOOD AT GIT

Understanding basic terminology

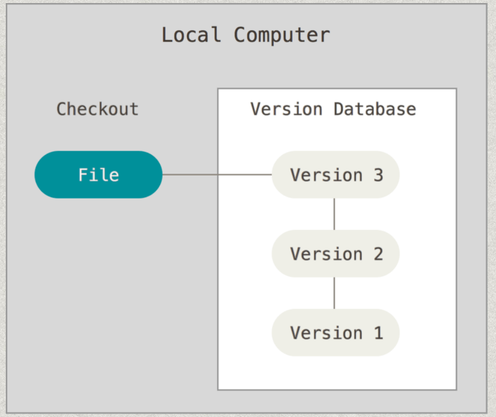
- File - a singular unit in a file system to store information.
- Directory - a file system term to refer to a structured list of files and other directories located somewhere in memory
- Git Repository - very similar to a normal directory only that it contains a special directory called the “.git” directory. Inside the “.git” folder is the repositories’ version database and is what the “git” software program uses internally.
- Local - an adjective, usually to refer when something is only available to a specific network or single computer.
- Remote - an adjective, usually to refer when something is hosted on a network.

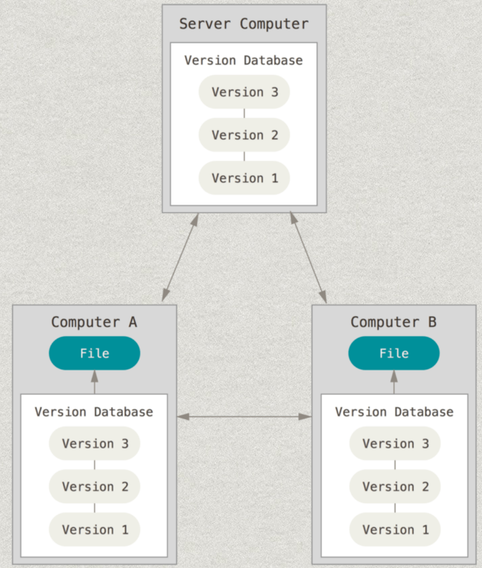




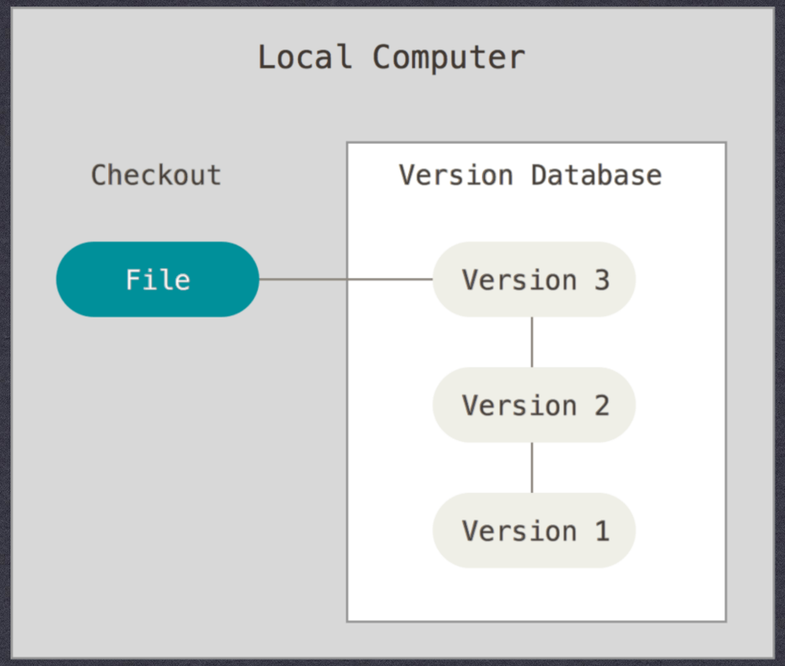
Understanding basic terminology

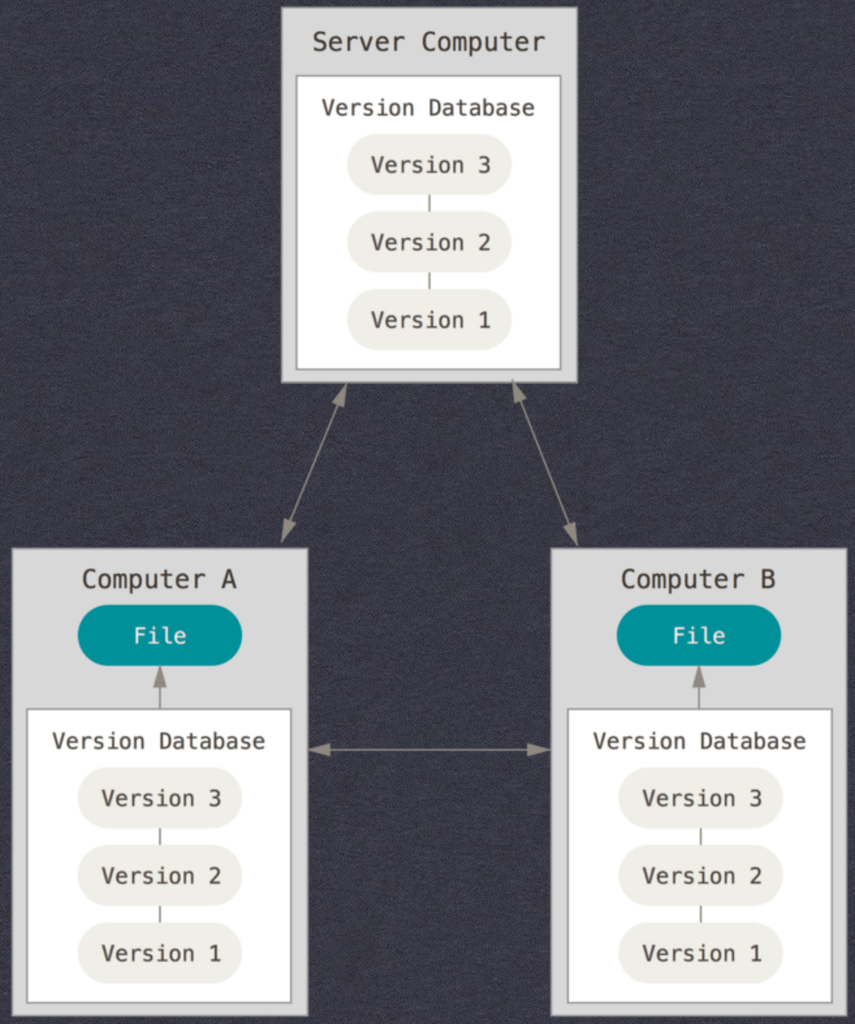
- Local Repository - a local version of a repository may it be a clone of a remote repository or an offline repository.
- Remote Repository - a remote version of a repository may it be a fork or a clone of an existing local or remote repository.
- Github Repository - a remote git repository hosted on Github’s servers.



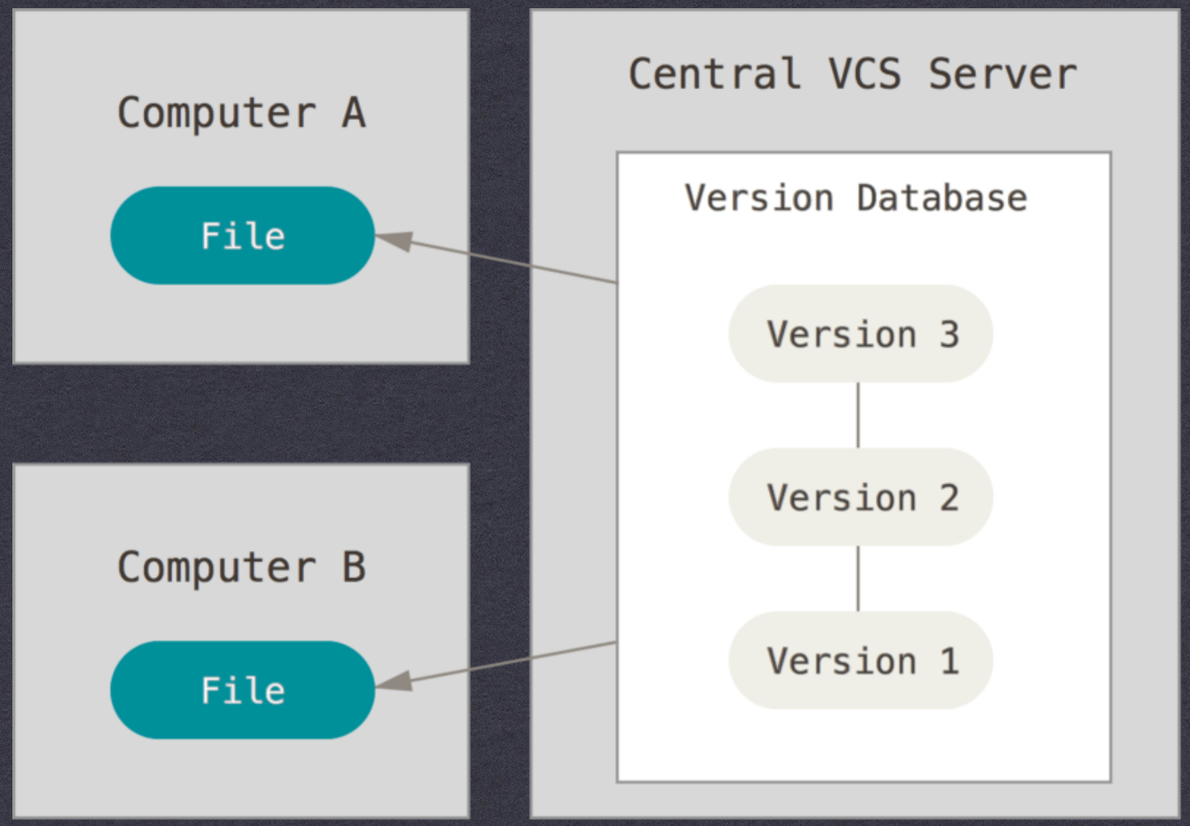


DISTRIBUTED REPOSITORIES LIKE GIT



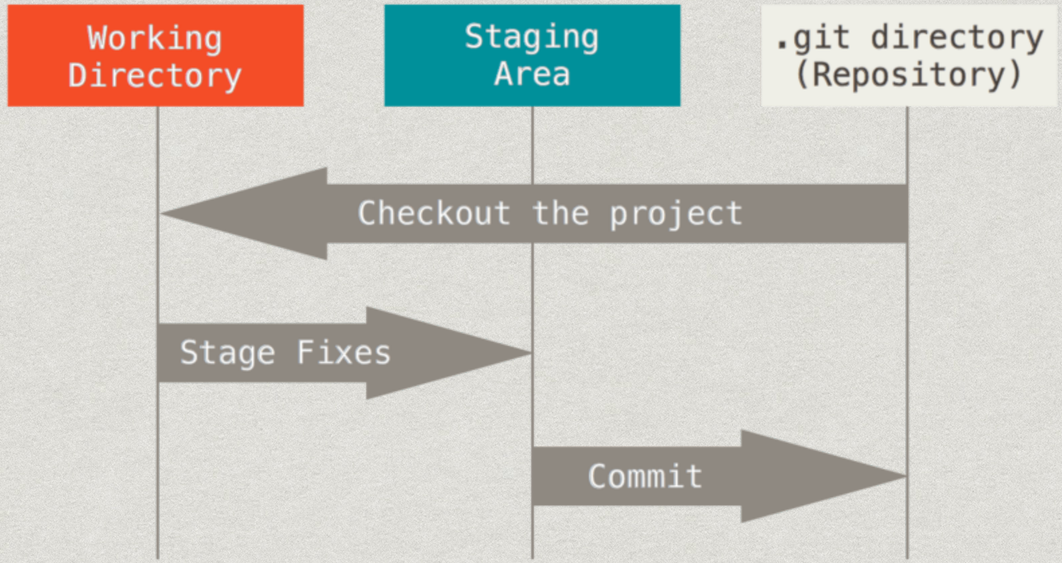


HOW OTHER NON-DISTRIBUTED CENTRALIZED VCS' WORK

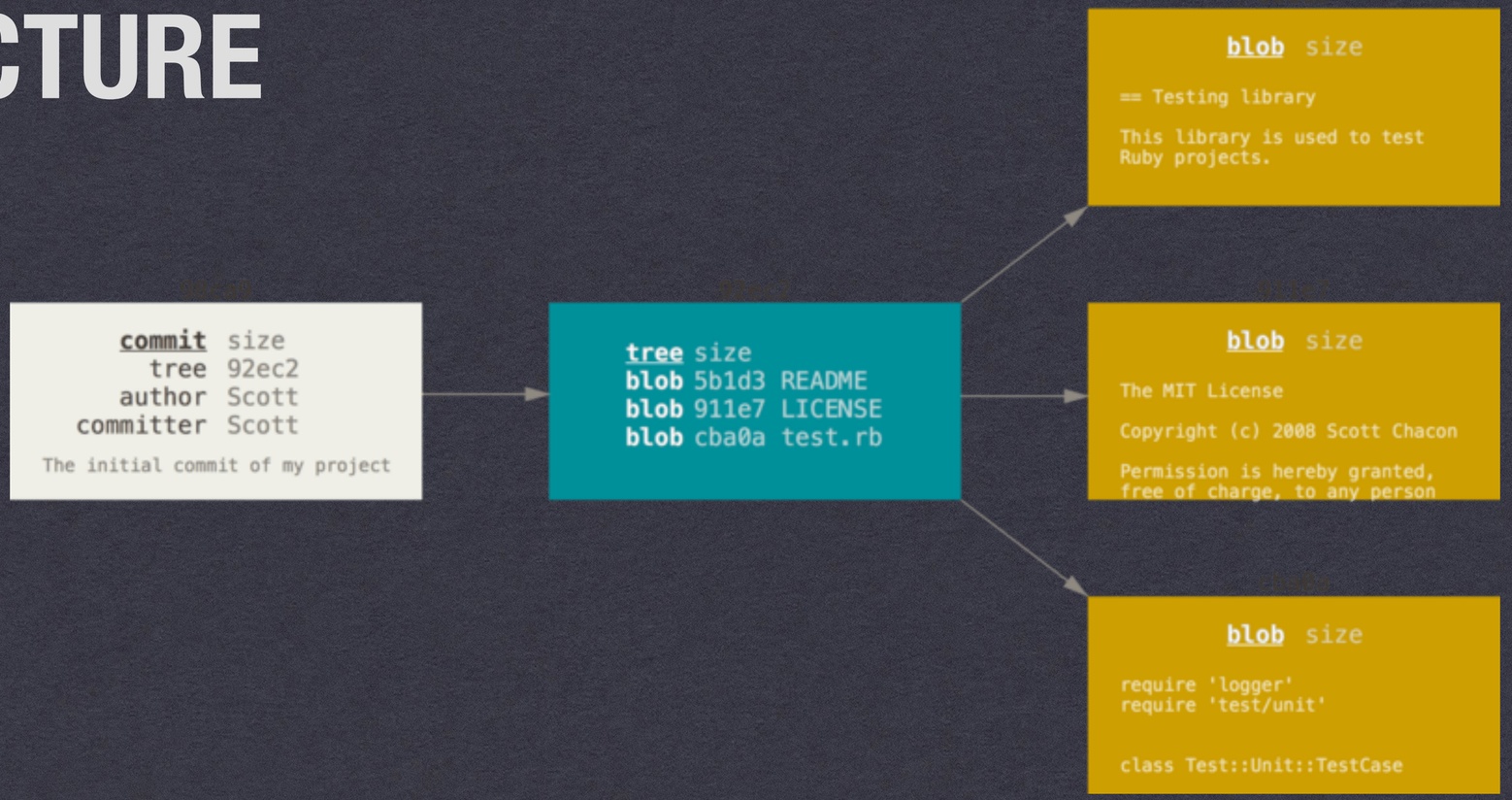


Understanding basic terminology

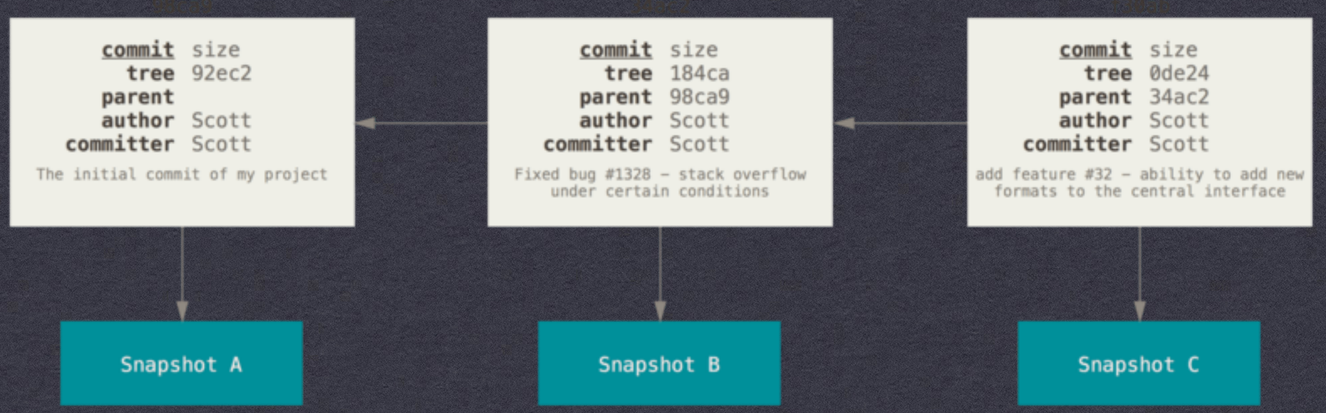
- Staging Area - a list of “modifications, additions, deletions”, where the changes are tracked and ready to be committed onto a branch.
- Working Directory - the location where the project files and directories reside.
- Branches - a pointer to a collection of snapshots (states, versions) of the project.
- Checkout - checkout means to “switch” the Working Directories’ state to match that of the desired branch.
- Committing - to append a new state of the Working Directory to a desired branch.



GIT COMMIT STRUCTURE

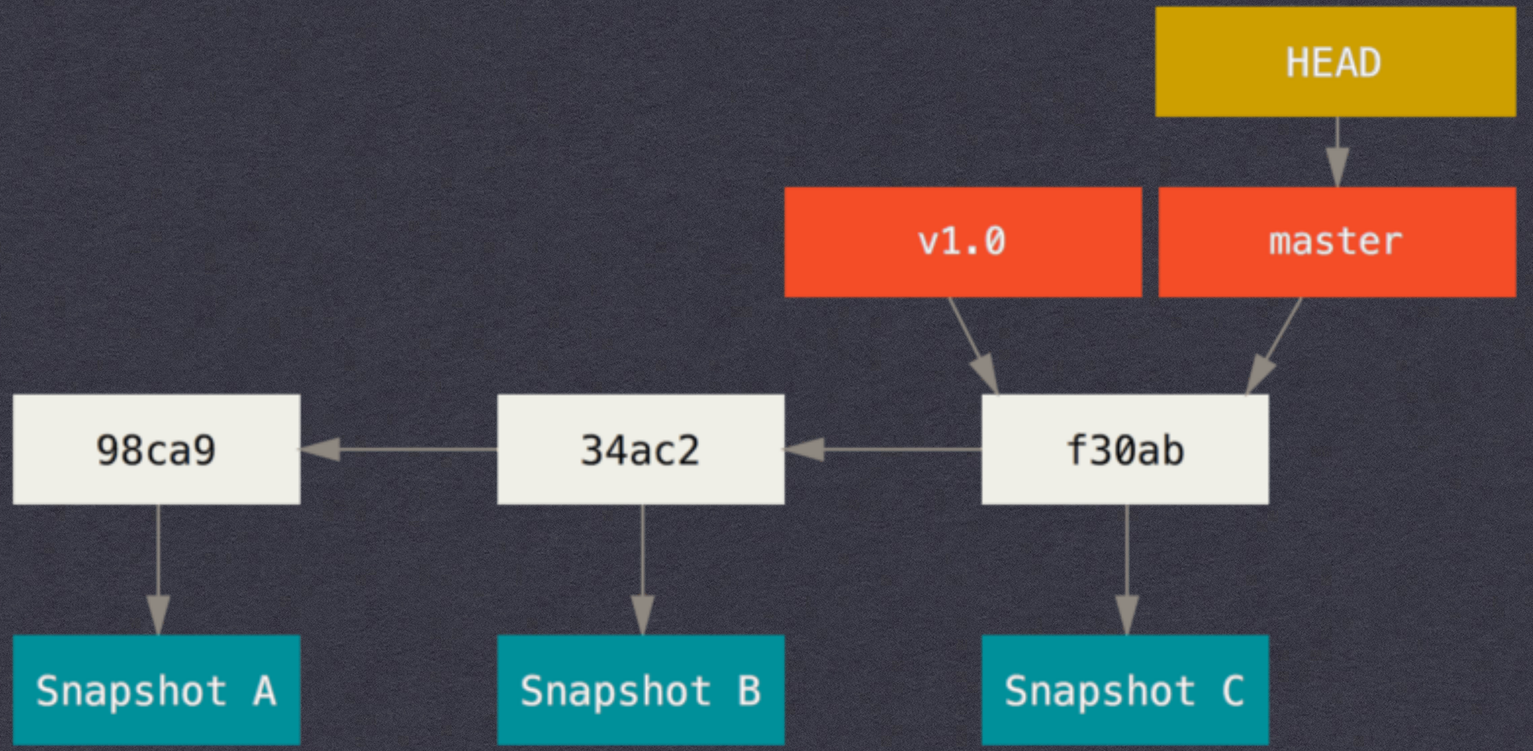


GIT COMMIT HISTOGRAM



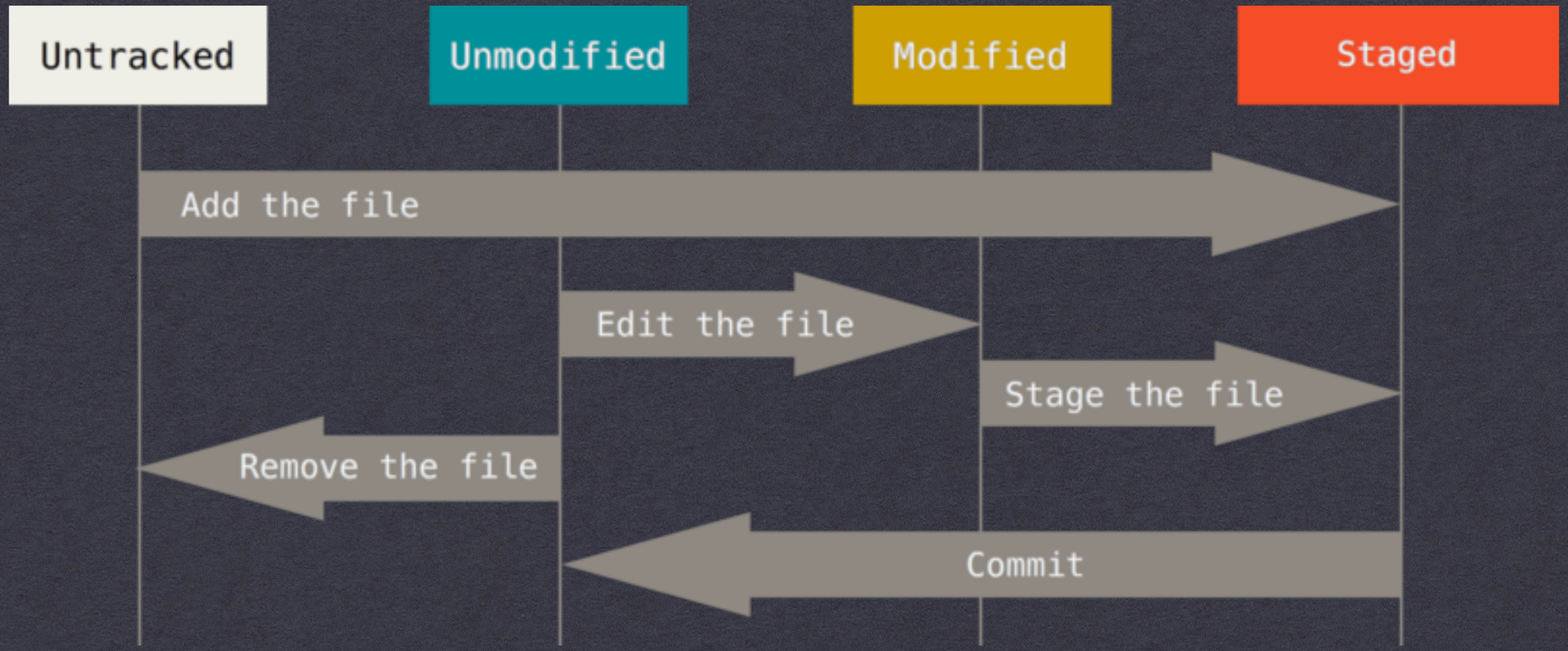
GIT BRANCHES

ARE POINTERS



GIT FILE STATUSES

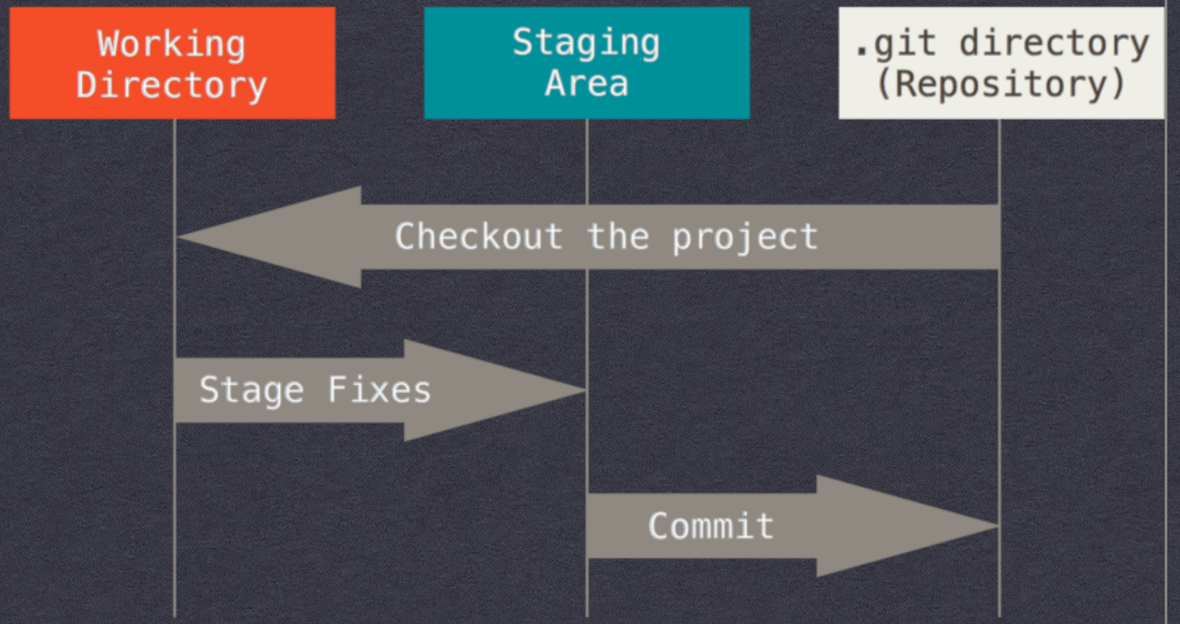
HELP YOU UNDERSTAND WHAT IS THE STATE OF YOUR PROJECT



GIT AREAS

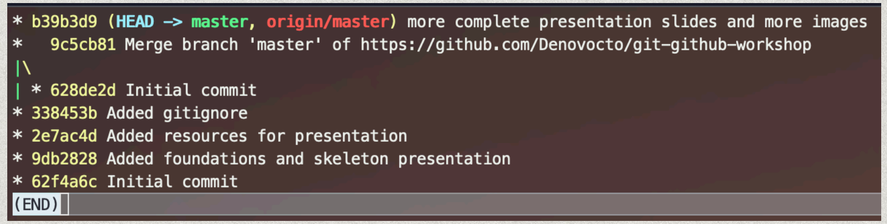
COMMIT PROCESS

- Standard Commit Process:
  - `git add <path>`
  - `git commit -m <message>`
  - `git push <remote>(origin) <branch>`
- Shorthand:
  - `git commit -am <message>`
  - `git push`



Git logs and checking the status of your project

- `git log --graph --decorate --oneline`
- `git status`



Git branch

- You can create branches with:
  - `git branch <name>`
  - `git checkout -b <name>`
- You can list fetched branches with:
  - `git branch`

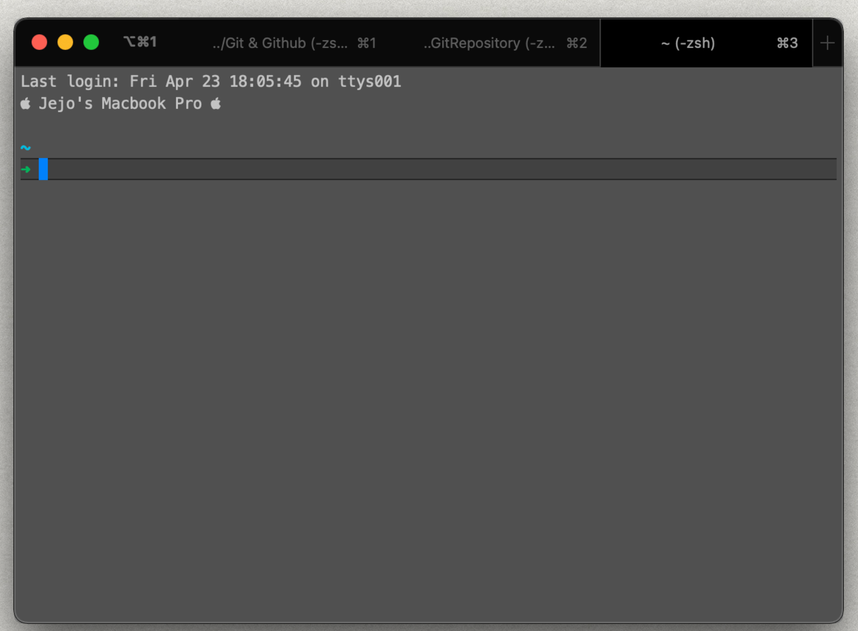
Git branch

- You can delete branches with:
  - `git branch -D <name>`
- You can checkout a branch with:
  - `git checkout <name>`

“Let there be init”

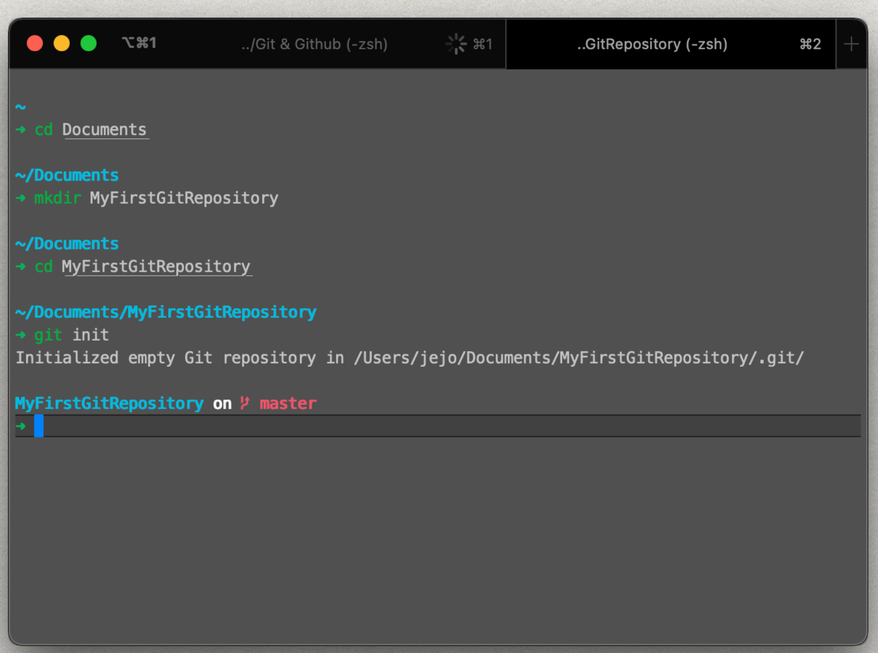
Creating your first repository

- For the purposes of this workshop and for better portability, we will be using a terminal application and it’s shell in order to interact with git.
- Windows: Powershell, Windows Terminal, Command Prompt.
- Unix/Unix-like: Any terminal application and shell.



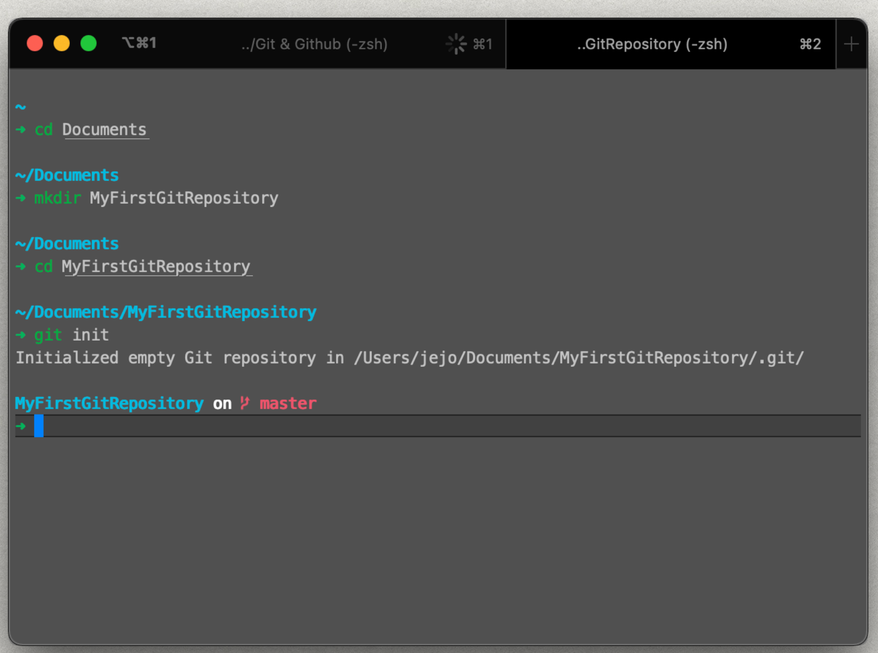
Creating your first repository

- Using the shell command “cd” short for change directory, move your working directory to the location where you would like the git repository to reside.
  - `Example: cd Documents`
- Using the shell command “mkdir” short for make directory, create a new folder with the name you would want the repository to have.
  - `Example: mkdir MyFirstGitRepository`



Creating your first repository

- Using the shell command “cd” short for change directory, move your working directory to the folder we just created called “MyFirstGitRepository”
  - `Example: cd MyFirstGitRepository`
- Using the shell command “git” and its argument “init” short for initialize, you are telling git to initialize your working directory as a git repository.
  - `Example: git init`
- Congratulations you have set up your first git repository.



Adding content & pushing to remote

- Use your favorite editor to write a file
- Create a designated remote repository through a repository hosting platform. (Github)
- Copy reference to that remote repository
- `git remote add origin <url>`
- Edit, Version, Commit, Etc..
- `git push --set-upstream origin master`

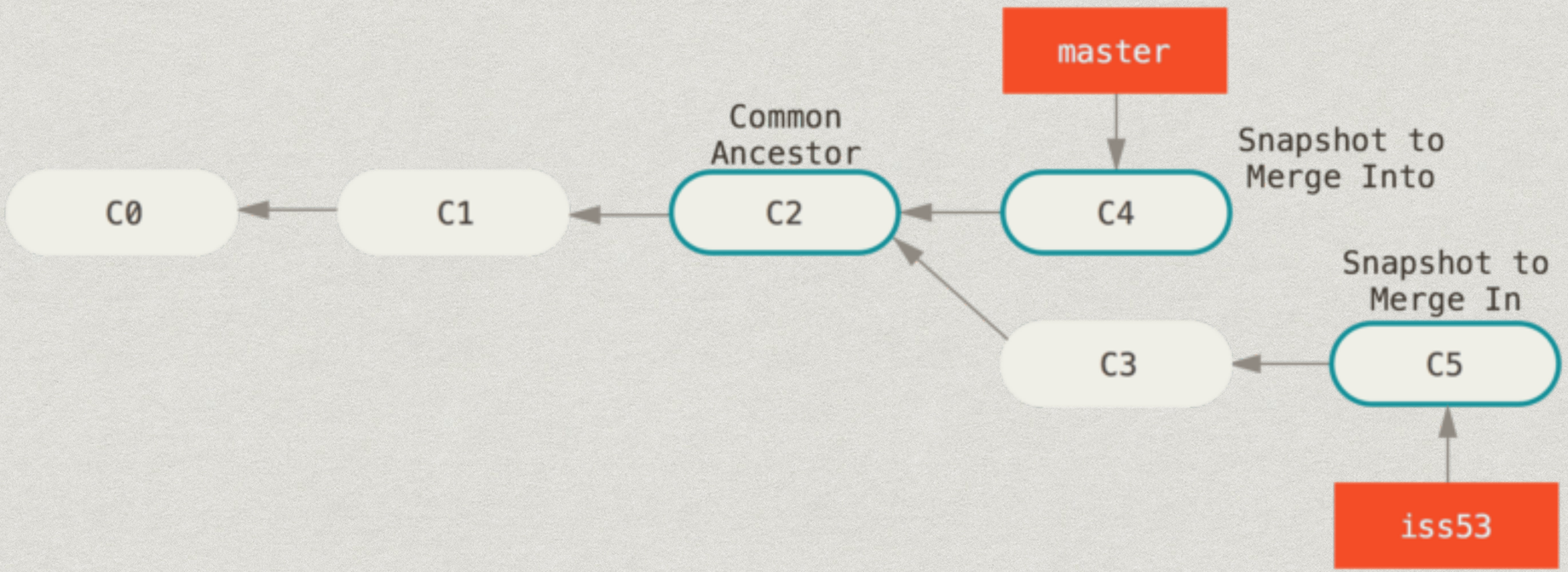
COLLABORATING WITH GITHUB

Github Repositories

- Creating a repository through GitHub
- Cloning and pulling from that repository
- Committing and pushing to that repository

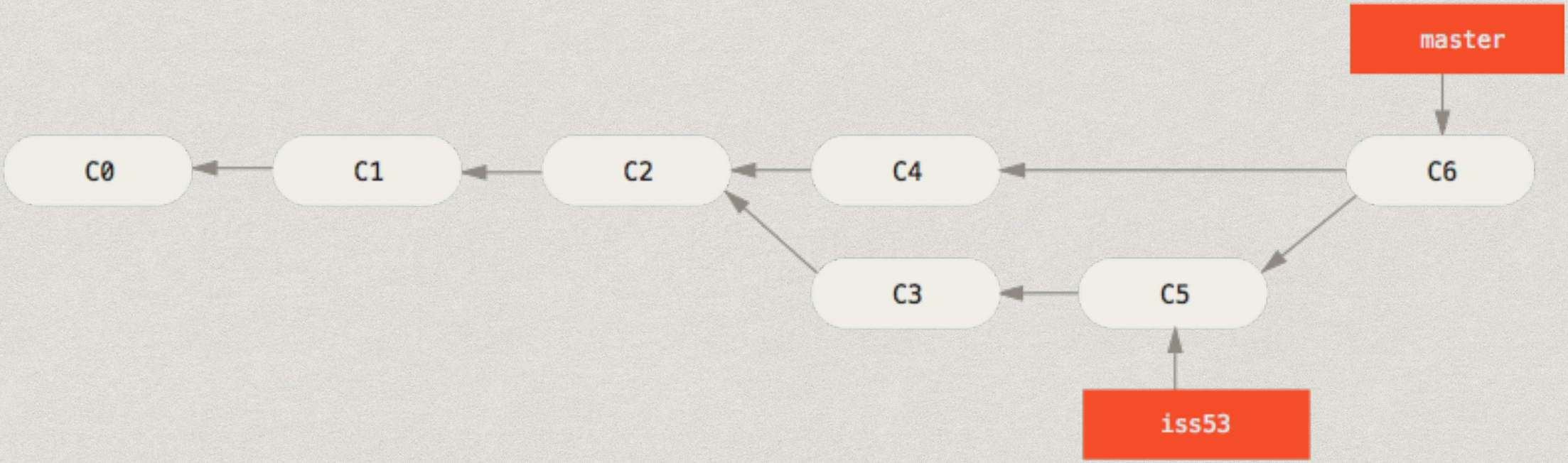
GETTING YOUR BRANCHES UP-TO-DATE

MERGING



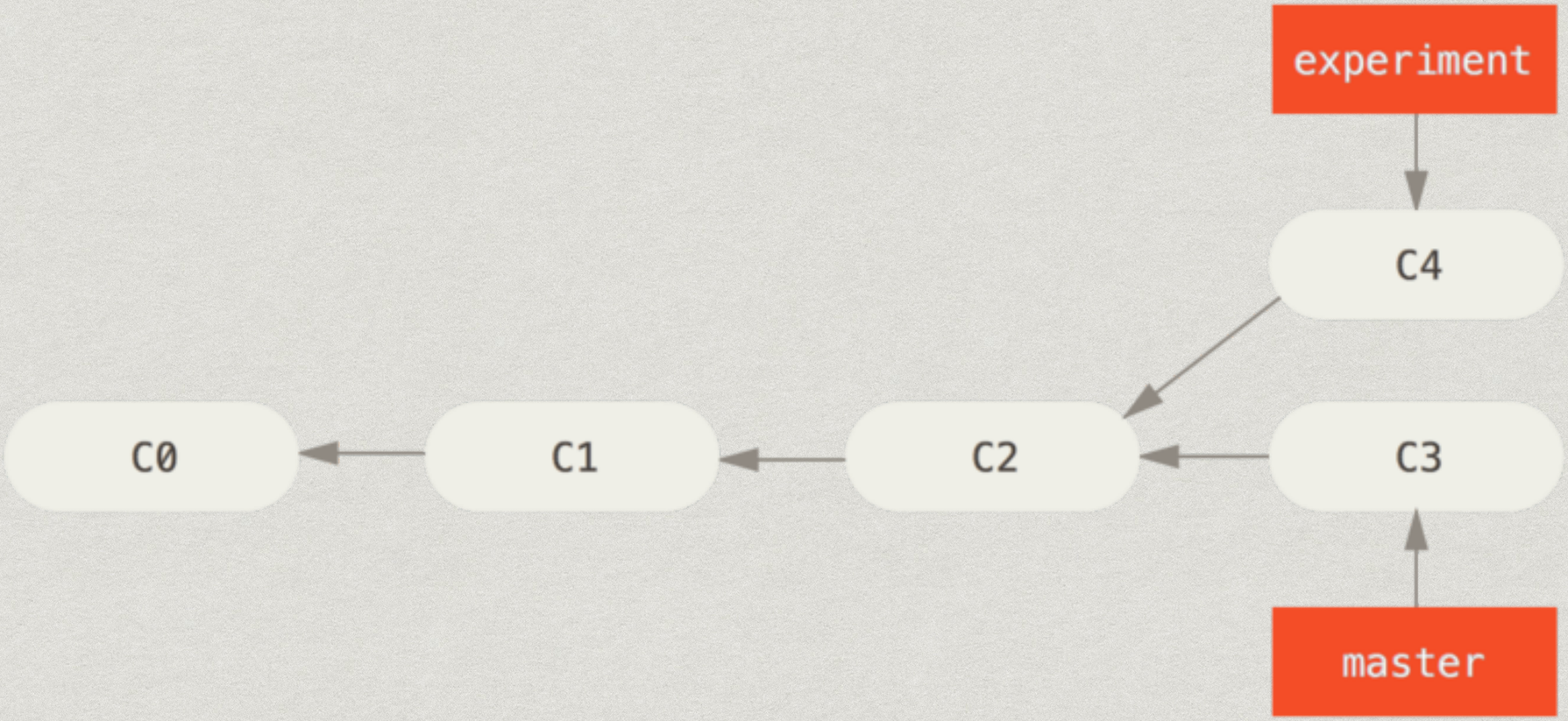
GETTING YOUR BRANCHES UP-TO-DATE

MERGING



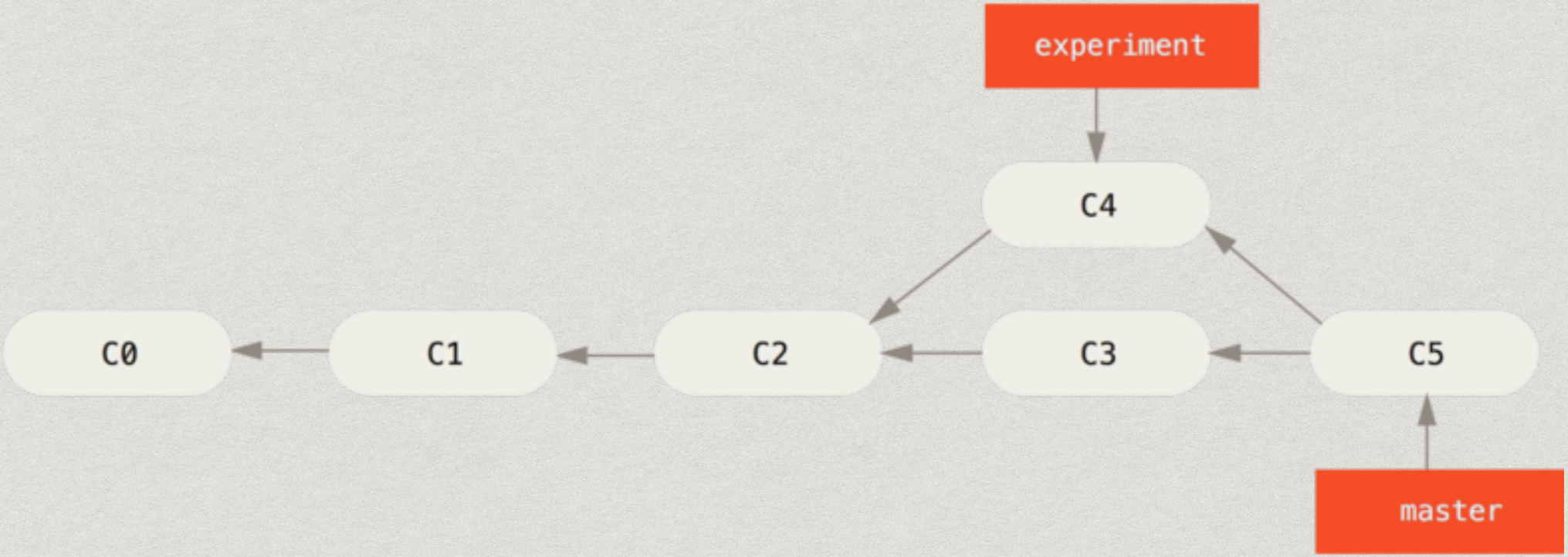
GETTING YOUR BRANCHES UP-TO-DATE

REBASING



GETTING YOUR BRANCHES UP-TO-DATE

REBASING

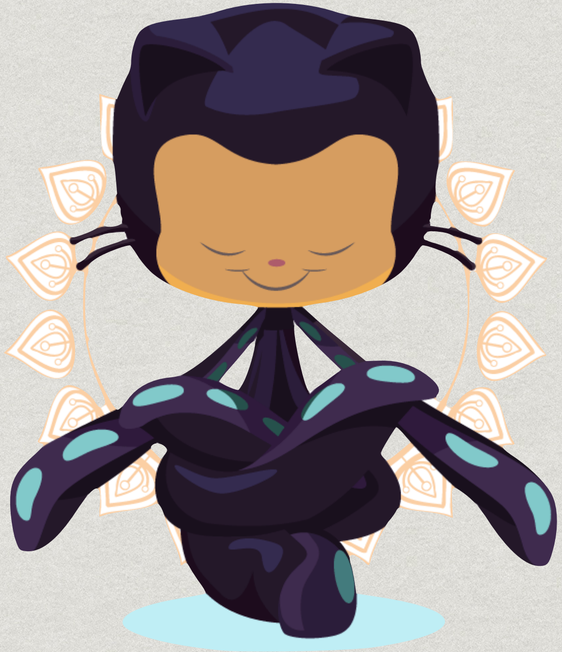


On Resolving Conflicts

- When merging, or rebasing, you may encounter a conflict
- Conflicts occur when git state machine detects two differing versions of the same file in the same locations when attempting to merge histories
- Conflicts can be avoided by maintaining branches up to date with master, feature or development branches, before committing newer histories

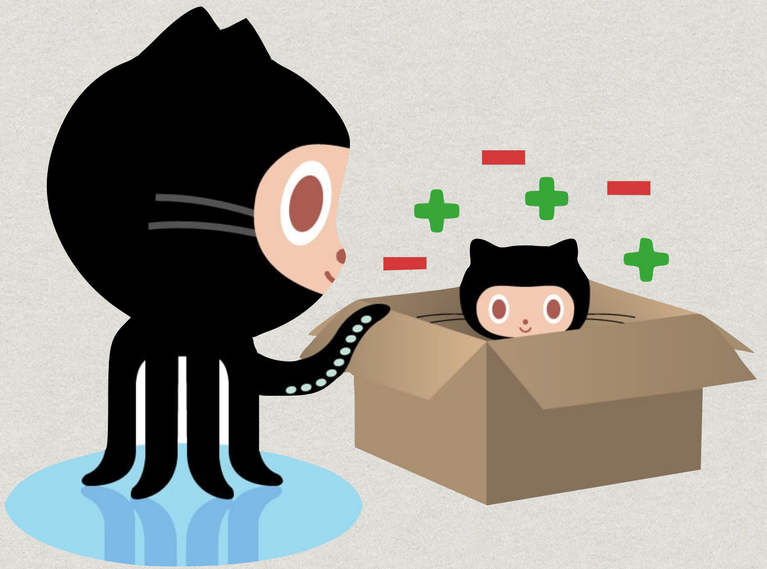
On Resolving Conflicts

- Conflicts can be circumvented by stashing working tree, merging, rebasing, or pulling and then popping or applying the stash
- Conflicts can be resolved manually and independently by reviewing differing versions and choosing what stays and what goes



Stashing

- Stashing is akin to committing, but instead of saving a snapshot of staging tree to a branch; it will save working tree to a stash which is branch-agnostic.
- Stashing is helpful when merging, rebasing, pulling and other special occasions
- To save a stash of working directory:
  - `git stash`
- To apply or pop stashes:
  - `git stash pop`
  - `git stash apply`



PULL REQUESTS WHAT ARE THEY FOR?

Conventions

- Common Conventions:
  - https://www.conventionalcommits.org/en/v1.0.0/
  - https://gist.github.com/digitaljhelms/4287848
  - https://docs.microsoft.com/en-us/azure/devops/repos/git/git-branching-guidance?view=azure-devops

TEMPLATING, NORMALIZATION AND CONTRIBUTION GUIDELINES

HTTPS://DOCS.GITHUB.COM/EN/COMMUNITIES

SSH AUTHORIZATION FOR GITHUB FOLLOW

HTTPS://DOCS.GITHUB.COM/EN/AUTHENTICATION/CONNECTING-TO-GITHUB-WITH-SSH