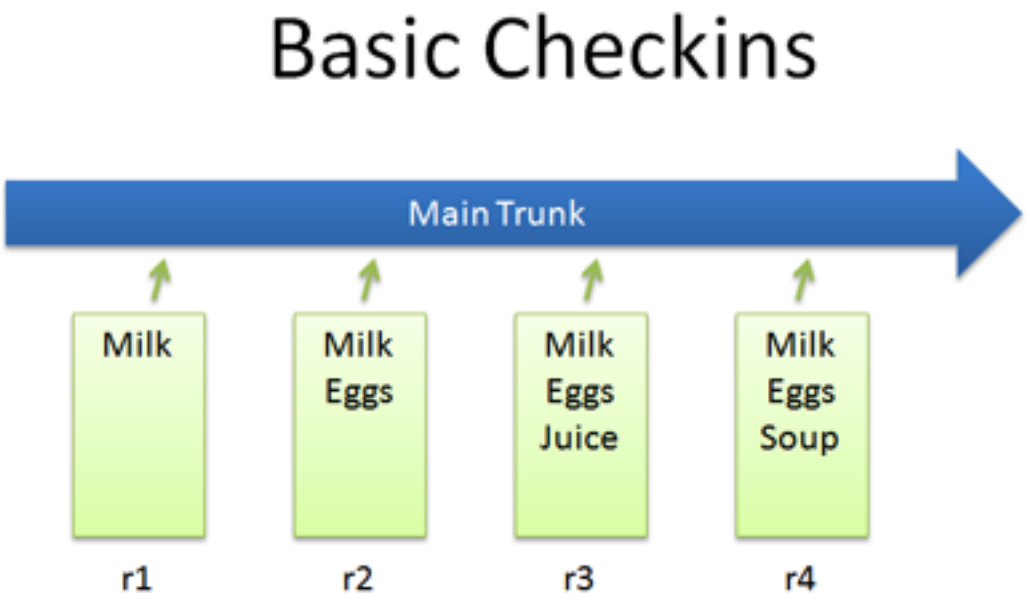
Git and GitHub

Version Control

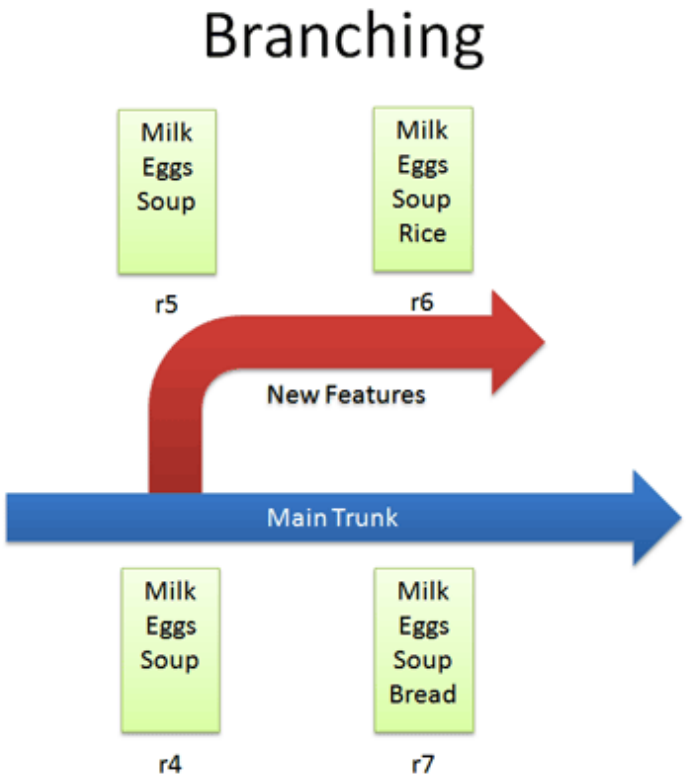
- Software that tracks changes to a file (or files) over time
- Modern software version control systems track not just the change itself, but who made it and when
- Enables (more organized) collaborative software development
- Repository – the group of files being tracked
- Why?
  - Roll back changes – infinite level of undo
  - Compare changes from the past to the present
  - Share changes among developers
  - See who last modified something that might be causing a problem and correct it

- Can update a file each time it is changed



Branching

- Make your own local copy to save and test changes



Branching

- Helpful for multiple individuals to do development
  - Don't overwrite each others code
  - Can introduce mistakes while developing without impacting the main working code base
  - Eventually check your clean working code back into the main project

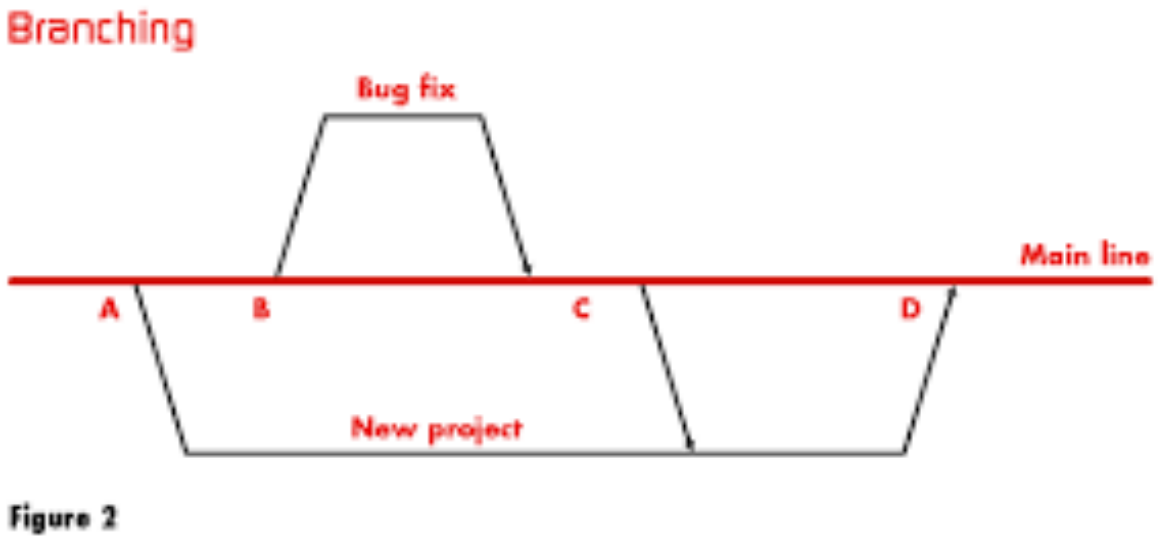
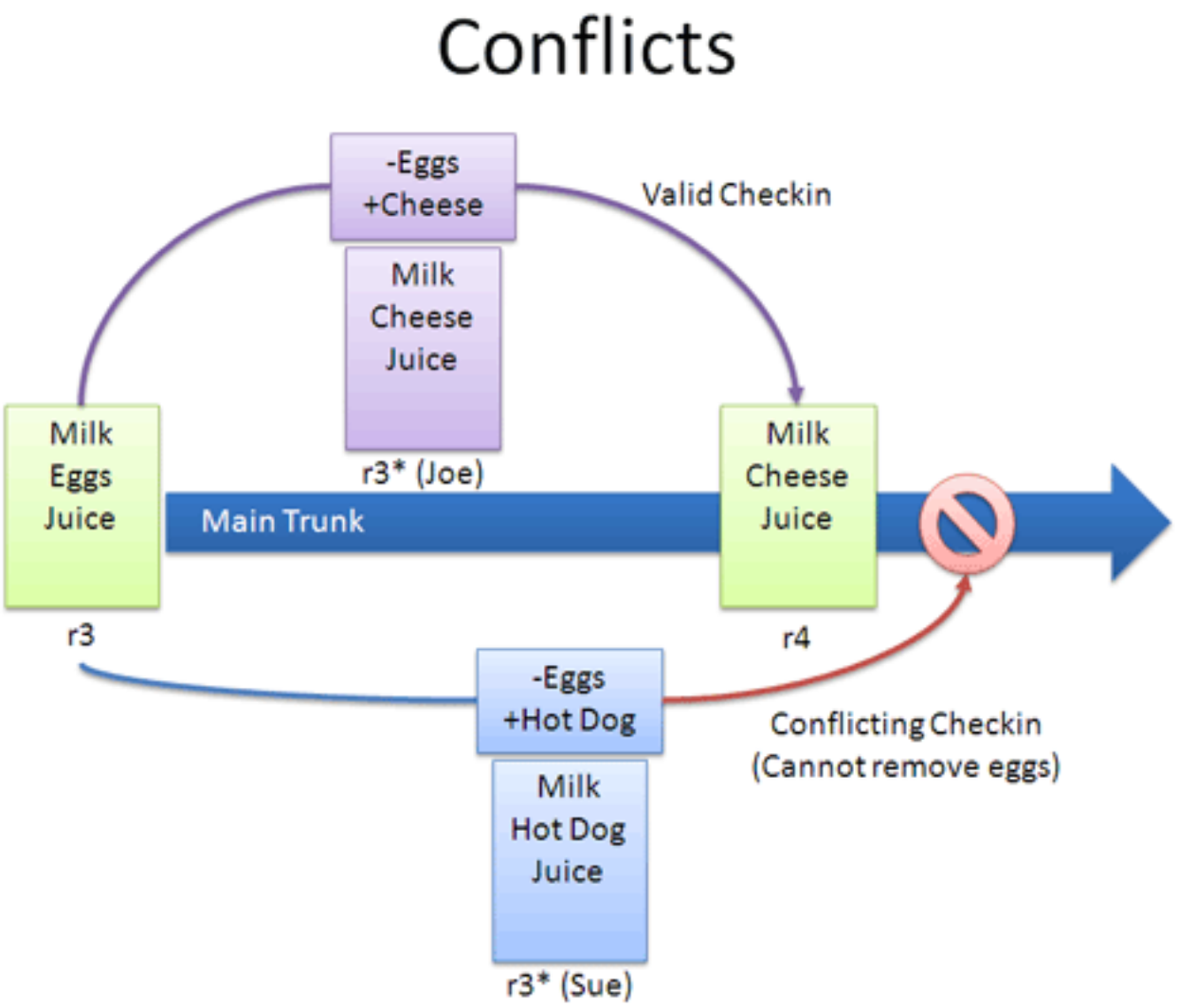


Figure 2

Merging

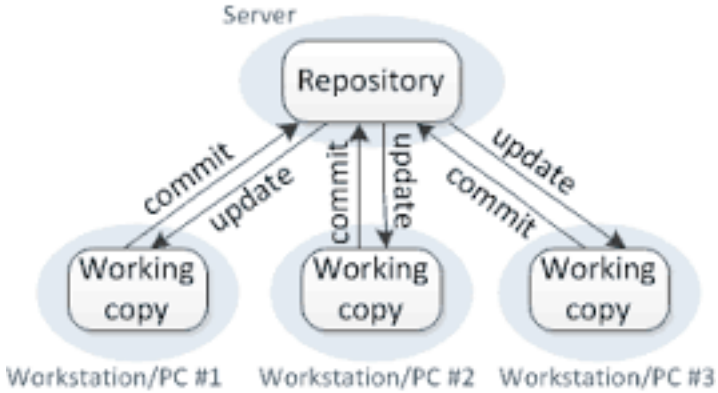
- You eventually want to merge your changes back into the central branch
- What if there are conflicts???
- A user must decide how to resolve



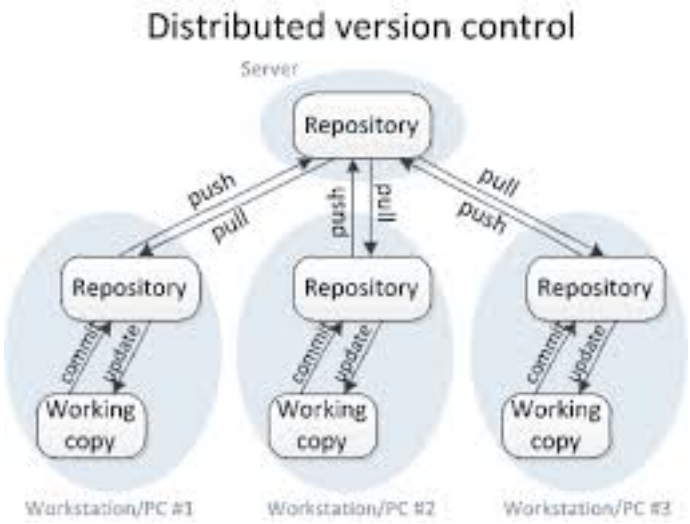
Git

- A version control system
- A *distributed* version control system
  - allows multiple systems to host entire copies of the repository
  - allows the users on those systems to collaborate on changes to the repository

Centralized version control



Distributed version control



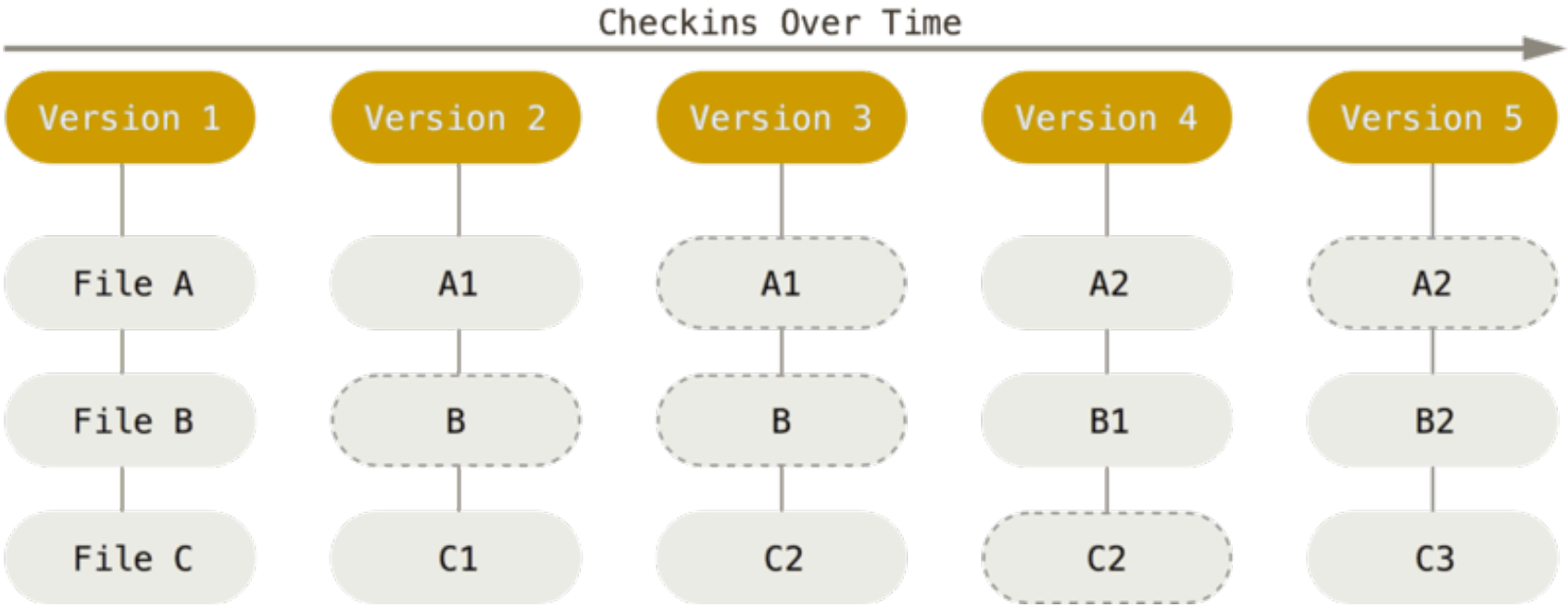
https://homes.cs.washington.edu/~mernst/advice/version-control.html

History of Git

- Created to manage the Linux kernel development in 2005 by Linus Torvalds
- Needed strong support for non-linear development (thousands of parallel branches and ability to handle large projects like the Linux kernel efficiently (speed and data size)
- Difficult learning curve - a bit complex to deal with these needs

Git fundamentals

Git thinks of its data like a set of snapshots of a miniature file system. Every time you commit, or save the state of your project in Git, it basically takes a picture of what all your files look like at that moment and stores a reference to that snapshot.



Git Fundamentals

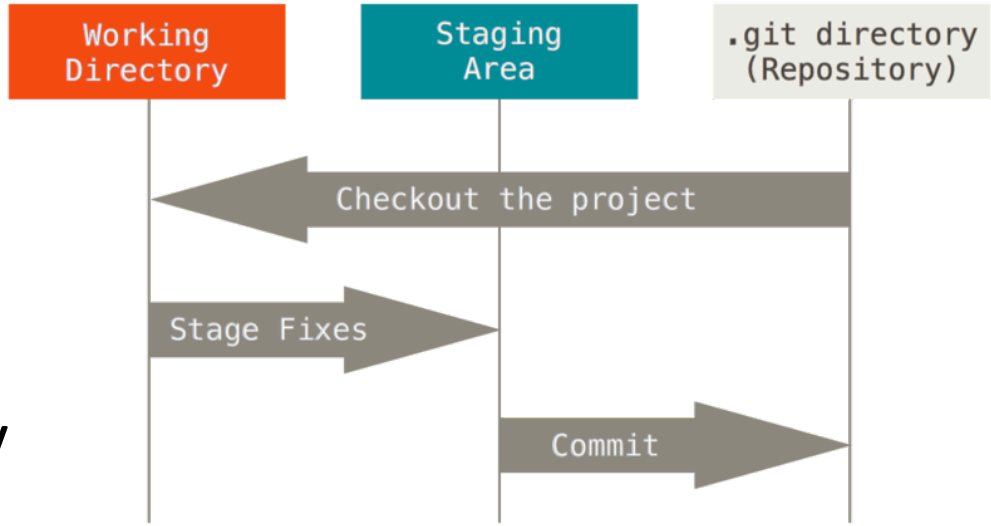
The Three States

Git has three main states that your files can reside in

- Modified means that you have changed the file but have not committed it to your database yet.
- Staged means that you have marked a modified file in its current version to go into your next commit snapshot.
- Committed means that the data is safely stored in your local database.

Git fundamentals

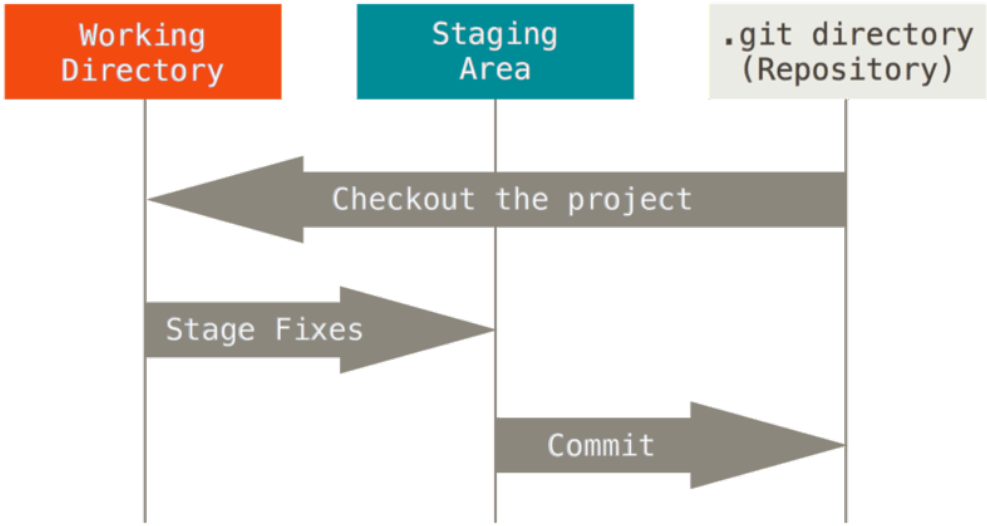
- The three file stages are reflected in the three main sections of a Git project:
  - the Git directory
  - the working directory
  - the staging area.

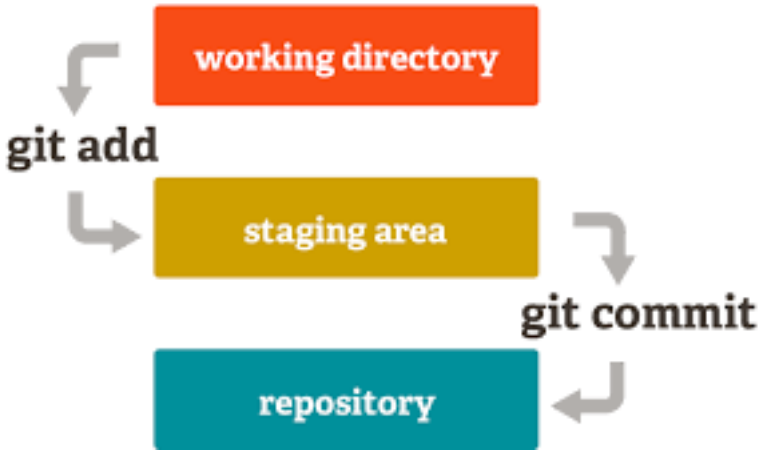


Git fundamentals

The basic Git workflow goes something like this:

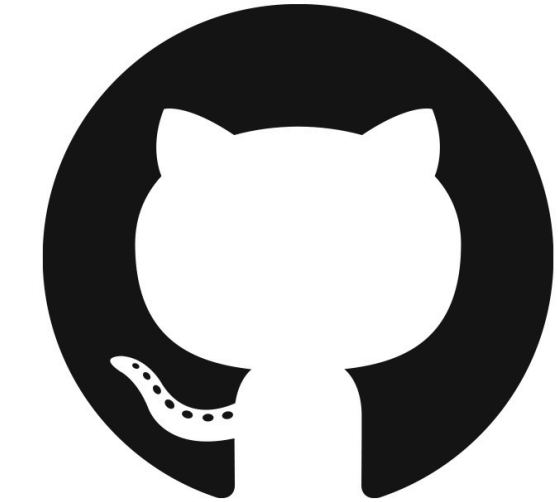
- You modify files in your working directory.
- You stage the files, adding snapshots of them to your staging area.
- You do a commit, which takes the files as they are in the staging area and stores that snapshot permanently to your Git directory.





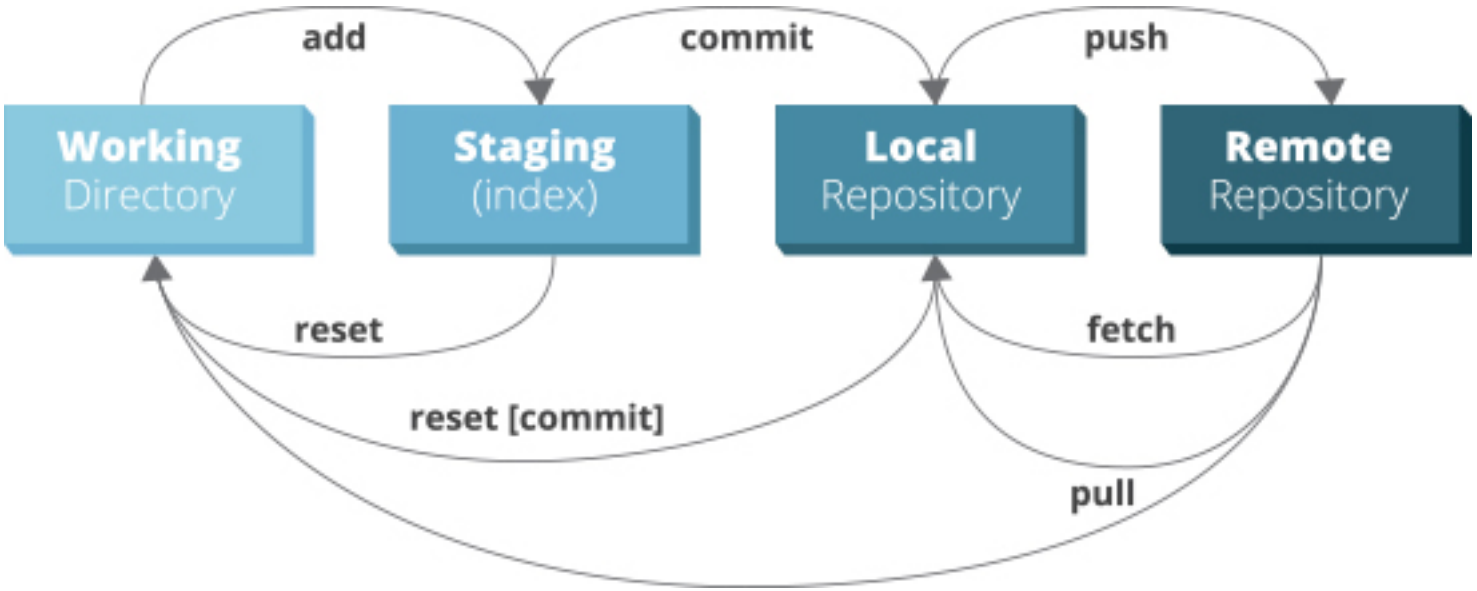
GitHub

- Web-base Git hosting service
- Like an online container for git repos
- (You can use other git-based online services such as BitBucket)
- Additional features
  - bug tracking, feature requests, task management, and wikis
  - Our course website!



Octocat

The complexity gets a bit worse...

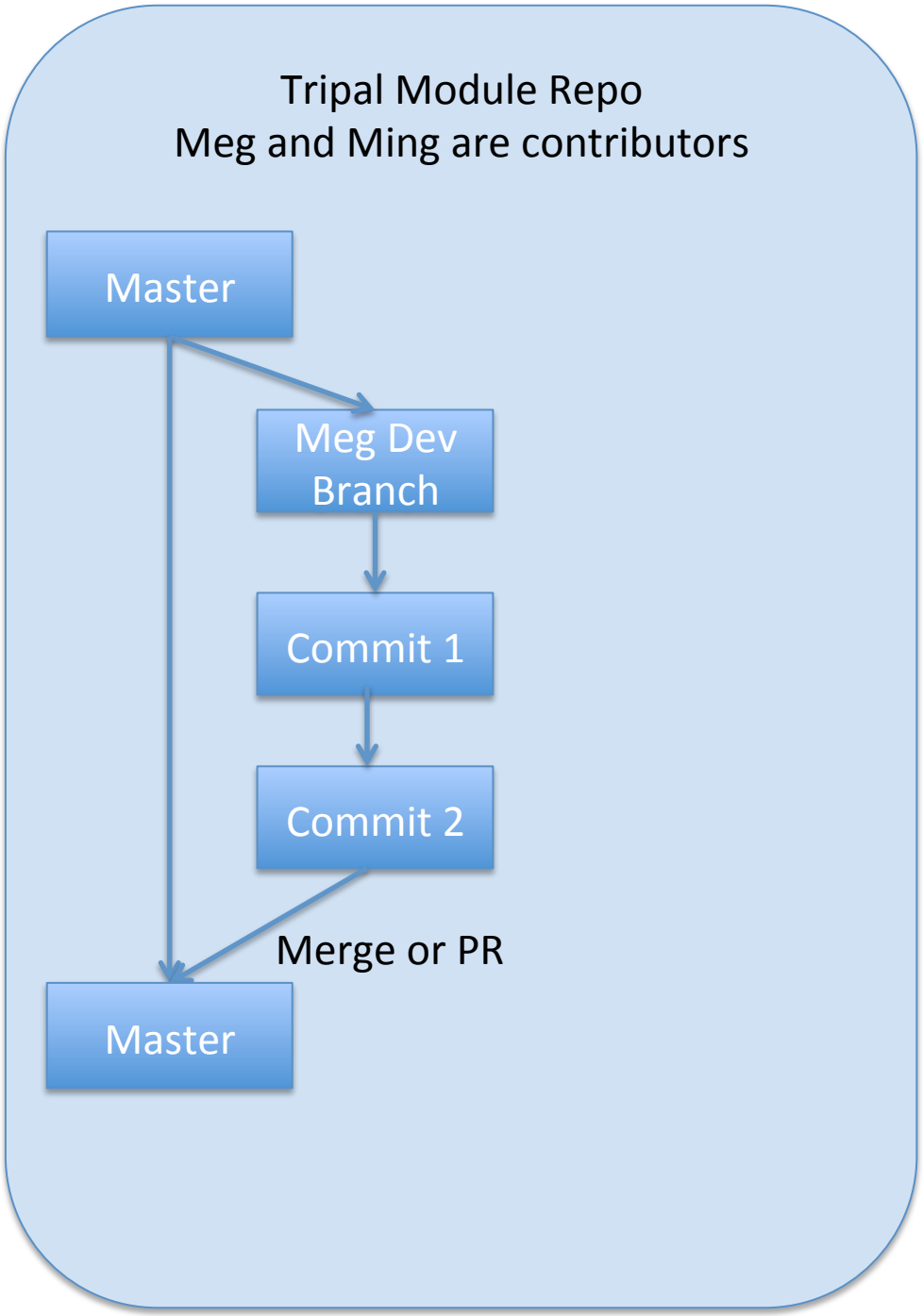


Tip: Search for a git cheatsheet and hang it on the wall. Forever.

Merge vs Pull Requests

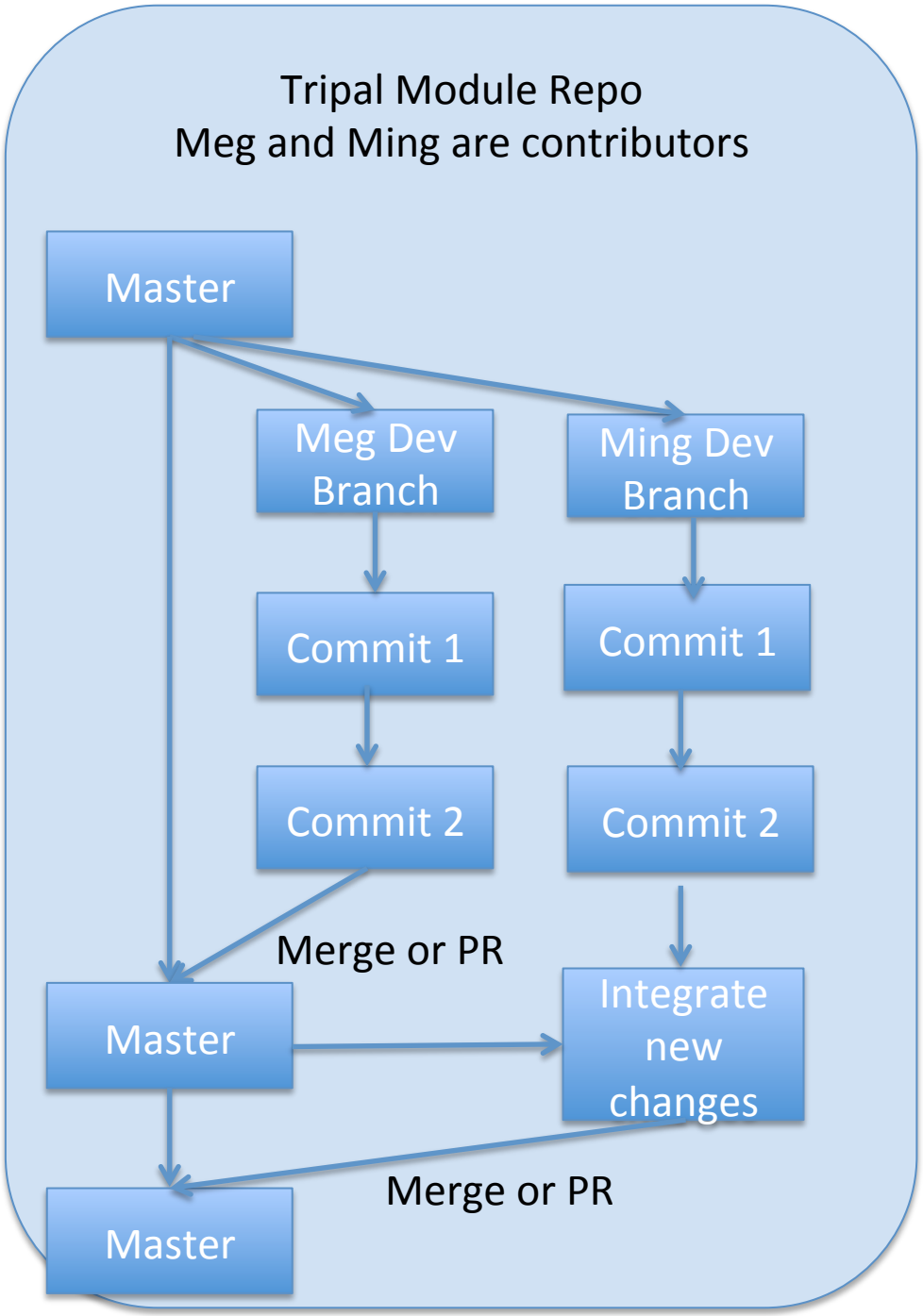
Two ways to make your branch changes part of the master branch:

- Merge
  - You must be a contributor
  - You are making the executive decision that this code is acceptable
  - Expedient if you are sole contributor or head honcho
- Pull request (From hosting service such as github)
  - Let others know about your changes
  - Contributors can discuss/review before accepting
  - A pull request can be created from a branch or from a fork



General:
https://guides.github.com/introduction/flow/

Can be far more comlex (and useful!):
https://www.atlassian.com/git/tutorials/comparing-workflows

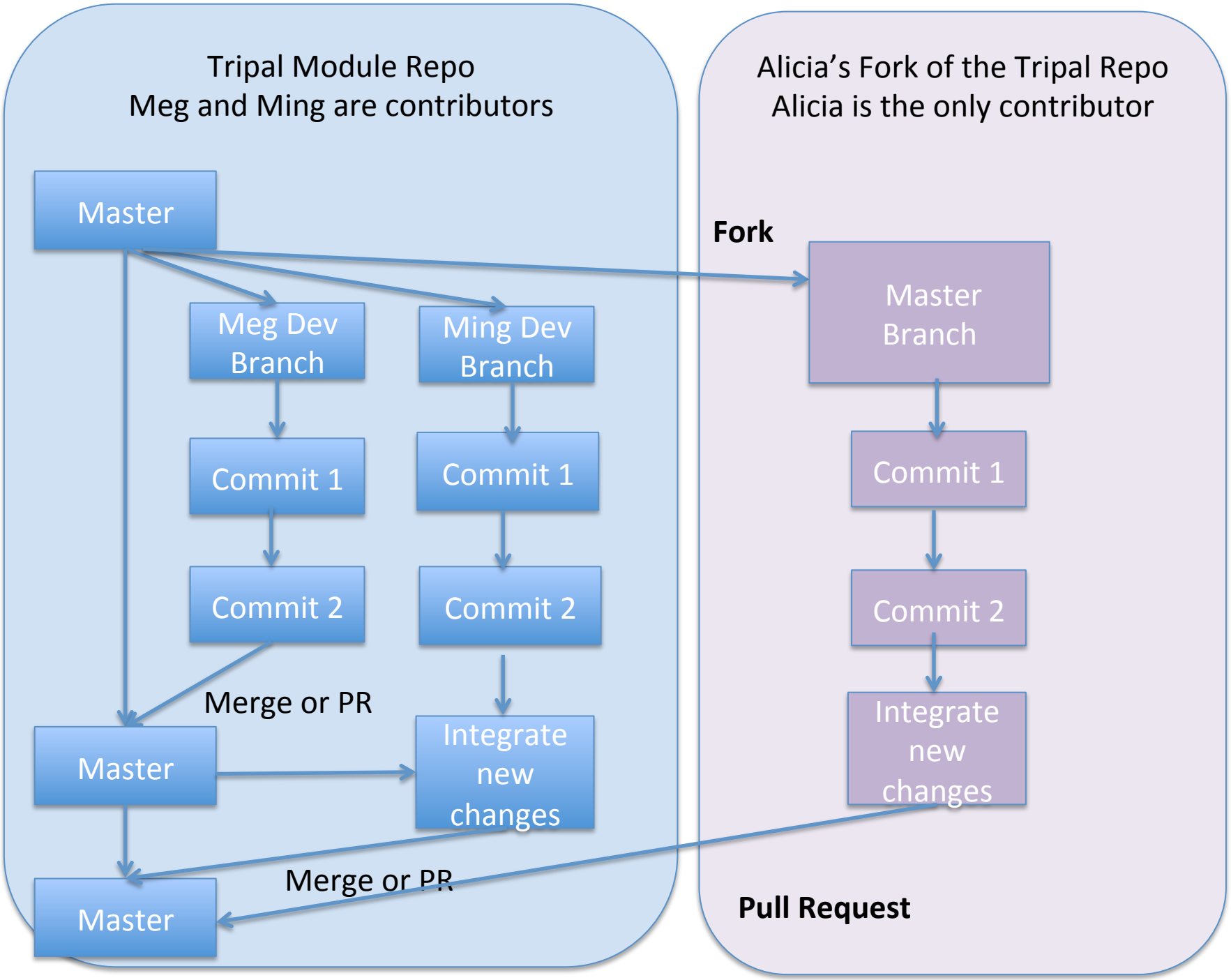


General:
https://guides.github.com/introduction/flow/

Can be far more comlex (and useful!):
https://www.atlassian.com/git/tutorials/comparing-workflows

Fork Vs Branch

- A branch is a lightweight concept – meant to continue work on an existing project
  - Only contributors can make commits, merges, and create branches
- A fork is making a whole new project, but basing it on an old project
  - This is your only option if you aren't a contributor to the repo
  - Good for starting development off of someone else's code
  - Crowd sourcing community development of code


Tripal Module Repo
Meg and Ming are contributors
Alicia's Fork of the Tripal Repo
Alicia is the only contributor
Master
Fork
Meg Dev Branch
Ming Dev Branch
Master Branch
Commit 1
Commit 1
Commit 1
Commit 2
Commit 2
Commit 2
Merge or PR
Integrate new changes
Integrate new changes
Master
Merge or PR
Master
Pull Request

Brief Demo

- Follow the Hello World Github Tutorial
  - Create a repo
  - Start a branch
  - Make changes to a file and push them to github
  - Open and merge a pull request
- Need a github account
- Go to github.com