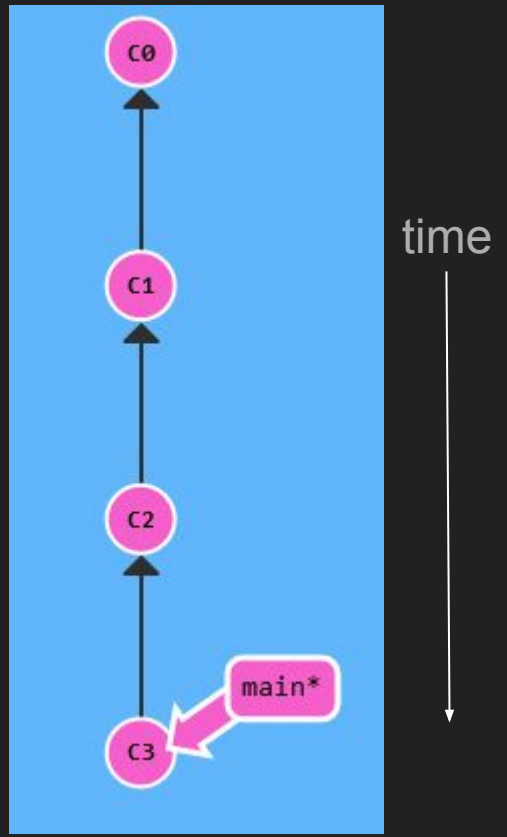
# Version Control

CMSE 890-602

# What is a *version*?

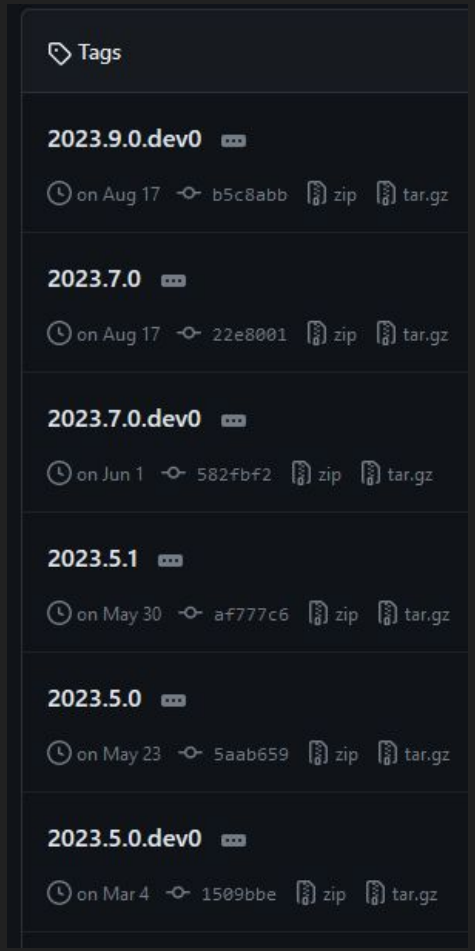
# Git definition of a version

- **A commit** referenced by a SHA-1 string (“hash”)
  - Contains all commit information
  - Refers to a *previous commit* hash (“parent commit”)
- Thus all *direct* version history can be found by following commits backwards in time



# Tagging versions

- Git allows you to apply tags to commits for easier reference
- This can act as a custom versioning system
- Tags can just point to a commit OR include a message and have their own hash
- Tags are often used for release markers



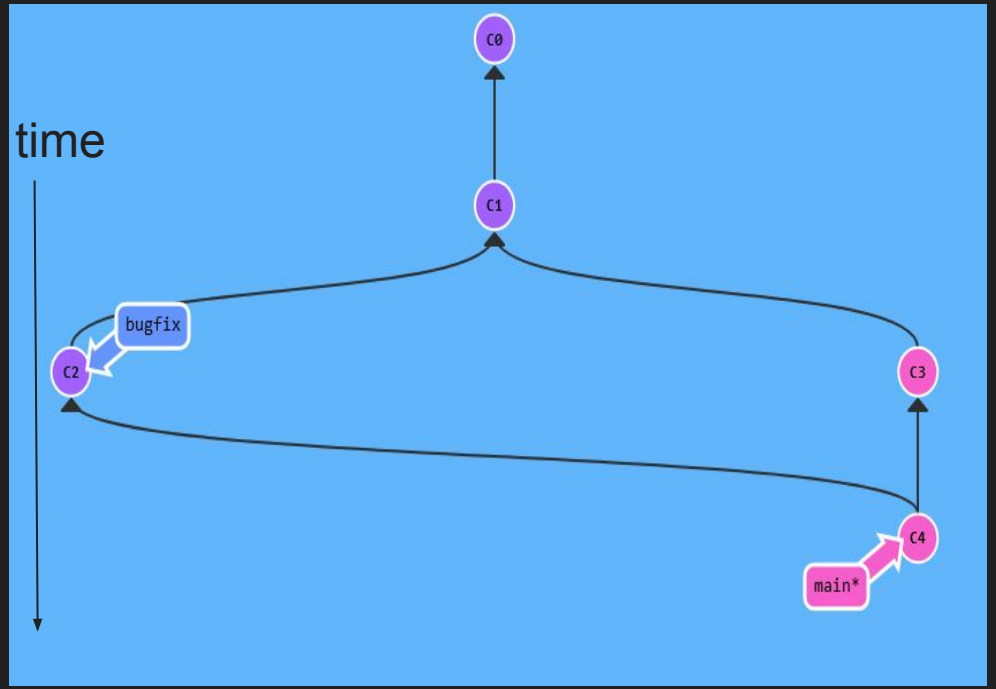
# Pre-class questions?

# Branches

- Branches have their *own* commit history that diverges from others at the point of creation
- Can be merged back into the main branch (sometimes called trunk)
  - Merge?
  - Squash?
  - Rebase?

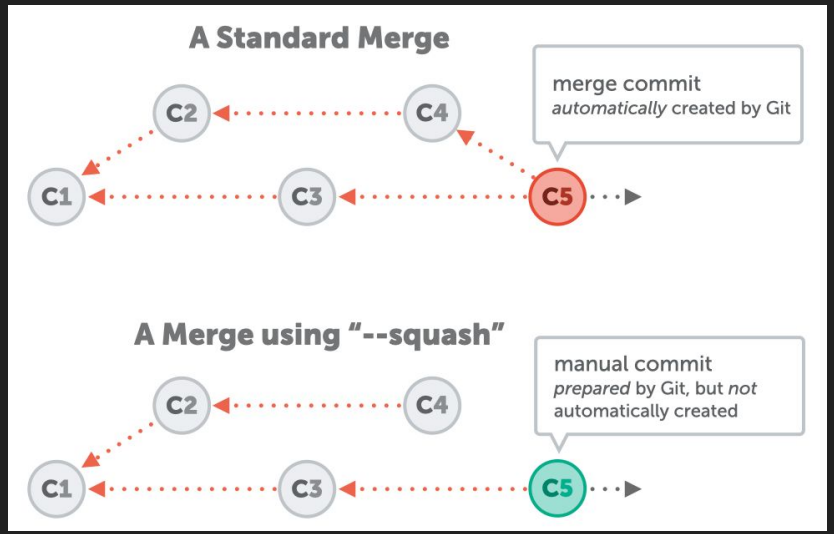
# Merge

- Takes commits from one branch and creates a merge commit pointing at those changes
  - and including the diff between the branches
- The new commit has *two* parents
- The branch commits become part of the main history
- If C3 was not made, C4 is not needed



# Squash

- Technically a kind of rebase
- Can be handled by GitHub on a merge
- Instead of a merge commit pointing at two parents the commits are *squashed* into one
- Branch history is *lost*
- Main history is simpler



# Rebase

- Reorganizes commits
- ***Rewrites history***
- Can be used to:
  - Reorder commits
  - Bring commits in from another branch
  - Remove commits
  - Squash commits together
- Can be horrible if there are conflicting changes between branches- these have to be handled per-commit

# Merge vs Squash vs Rebase

- Merge if the branch history is simple
- Squash if you want to keep main simple
- Rebase if branches have diverged significantly
  - OR if you want to clean up a branch history

# Forks

- Copies of a repository that can diverge
- Can be updated from the “original” repository or other forks
- Origin: your fork of a project
- Upstream: the original repository
  - Caution, upstream in git also refers to any remote repository
- Be careful of the license!

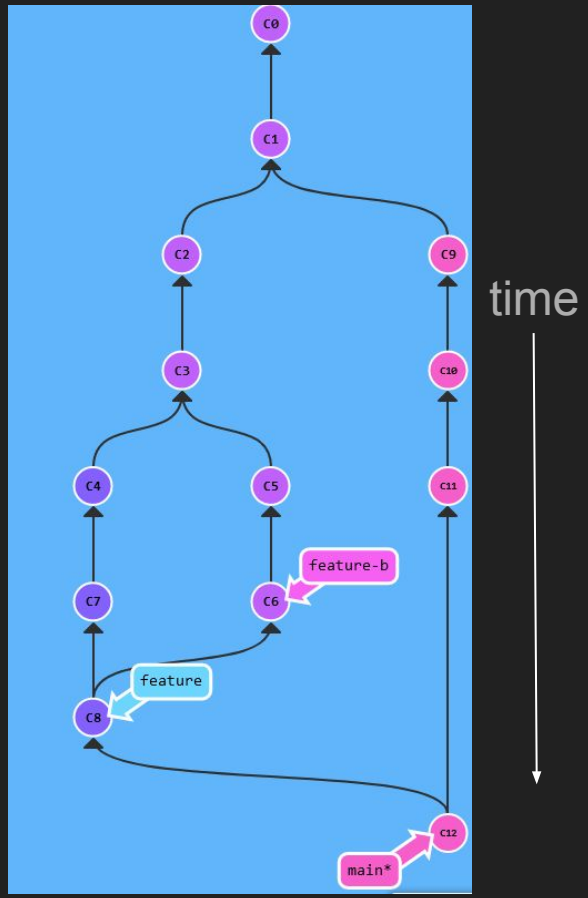
# Fork and branch workflow

1. Fork the repository
2. Create a new branch in your fork
3. Work in the branch
4. Create a pull request in the original repository

- Avoids permissions issues in the original repository
- Extra branches have no effect on the original repository
- Common in open source for external contributors

# Branch from a branch

- You can make as many branches as you want
- Branches do *not* have to be created from main
- Useful for working together on new features
- Can reduce merge conflicts into main
    - But may create conflicts on the branch itself



# Group Activity

- Open “advanced_git_in-class.pdf” from the main classroom repository
- Follow the instructions
- **Add me as a contributor to your repository**
  - andrewgfullard

Discuss your semester project with your table!

# Homework: get started on your semester project DFD

- Work in whatever software you feel comfortable with
- Follow the **symbol guides** from class 1
- Think about how to break your project into **small parts**
- Think about **reasonable** scope!
- Submit to me early for feedback if desired
- Take into consideration feedback from class 1
- Final submission due Oct 8th on D2L

# Pre-class for next week

Read https://docs.github.com/en/actions/learn-github-actions/understanding-github-actions.

Then, create and run the starter action in the classroom repository: https://classroom.github.com/a/chB0NDGr