CSE 3902: Git and GitHub

Justin Holewinski

The Ohio State University

Source Control Management

Source Control Management (SCM) is software that helps track changes

- Also known as *source control* or *version control*
- Records *snapshots* of your code at developer-determined intervals

Why?

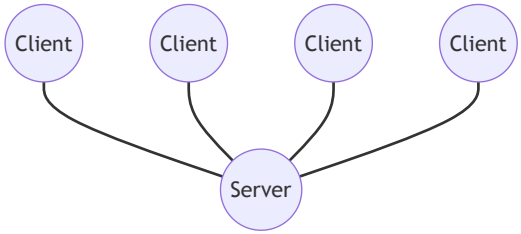
- Look at previous versions of the code
- See who made a particular change
- See why a particular change was made
- Share code with other developers in a coherent manner

Comes in two flavors:

- Centralized
- Decentralized

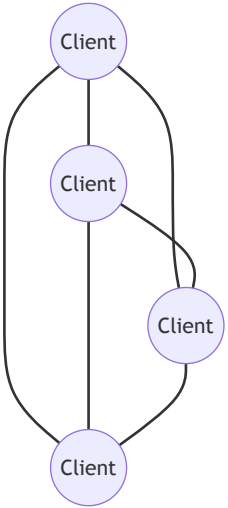
Centralized Source Control

- Standard client-server model
- Server is single-source-of-truth
- Clients communicate changes to server
- Examples
  - Subversion
  - Perforce (without DVCS)



Decentralized Source Control

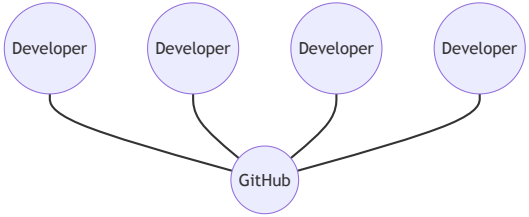
- Commonly known as Distributed Version Control System (DVCS)
- No server required, everyone can share changes with everyone else
- Examples
  - Git
  - Mercurial
  - Perforce DVCS



Decentralized Source Control

Common DVCS usage does have a centralized component

- Take advantage of both worlds
- Allow distributed code sharing *and* a centralized source-of-truth
- DVCS systems typically provide more features for the developer



Git

Advantages

- Very popular and always growing
- Lots of hosting services
  - GitHub
  - GitLab
  - Azure DevOps
  - BitBucket
- Lots of great features
  - Light-weight feature branches
  - Smart merging
  - Scalable

Disadvantages

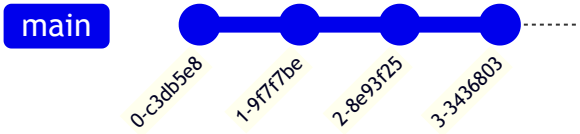
- Learning curve can be steep
- Not great with large binary files

Git Basics

Source history is maintained as a sequence of *commits*

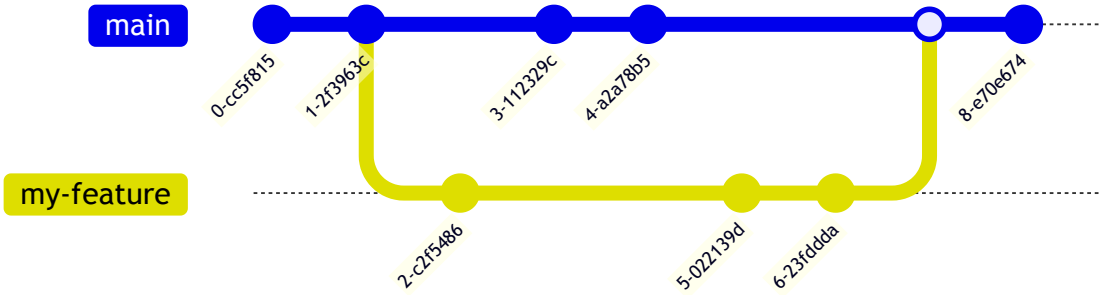
- Each commit is a "snapshot" of the code base at the time of commit
- Each commit is identified by a unique SHA hash
- Each commit is attributed to an author
- Each commit has zero or more predecessors

All of history is maintained as a DAG



Git Basics

New features are normally developed on *feature branches*



A *merge* commit joins two divergent code lines

Git Basics

Common git operations you'll be using:

- `clone`: Create a copy of another git tree
- `fetch`: Copy new changes from another git tree
- `push`: Copy changes to another git tree
- `merge`: Join two divergent code lines
- `pull`: Equivalent of `fetch` + `merge` (usually)
- `rebase`: "Replay" a branch on another branch

Git and GitHub Demos

- Creating a GitHub project
- Cloning a GitHub project
- Making a change and commiting to git
- Pushing changes to GitHub
- Opening a Pull Request
- Reviewing a Pull Request
- Merging a Pull Request
- Creating GitHub Issues