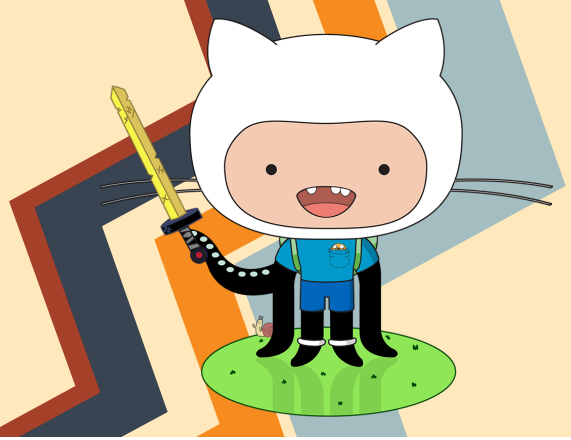
Forking, Pull Requests, Workflows and Rebasing

by Jonathan Miedel
and Alvin Wang



Last Week on Git

- Git tools
  - Specifically Merge Tools
- Git gc
- Git clean
- Git Stash

Review

- What are two major categories of merge tool?
- Name your favorite merge tool.
- What does git stash do?

Forking

- not an actual git command
- higher level concept implemented on GitHub and other repository hosting services
  - copies the repo on the server as a new repo under your username
- What limitation of Git makes forking a necessary step in many workflows (think back to our discussion on centralized version control systems)?

Problems with git merge

- creates complex tree structures
- merge commits
- lose effective commit history when merging branches

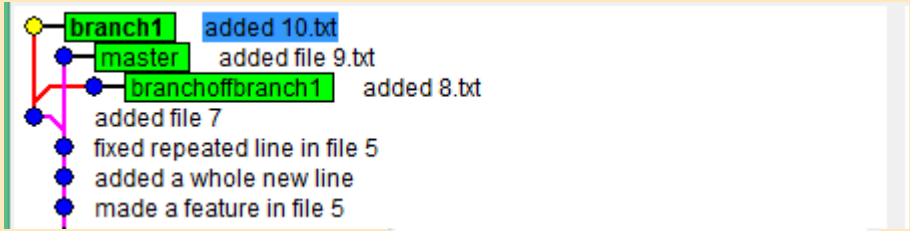
Rebasing

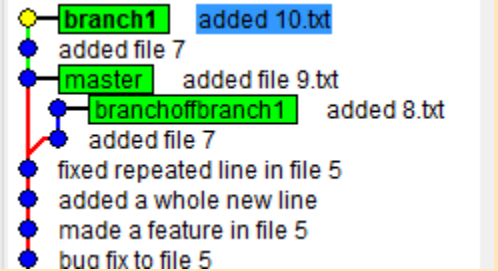
- git rebase
- Serves as another way to integrate changes from one branch to another
- Do not rebase public commits
- Replays the commits of one branch on another

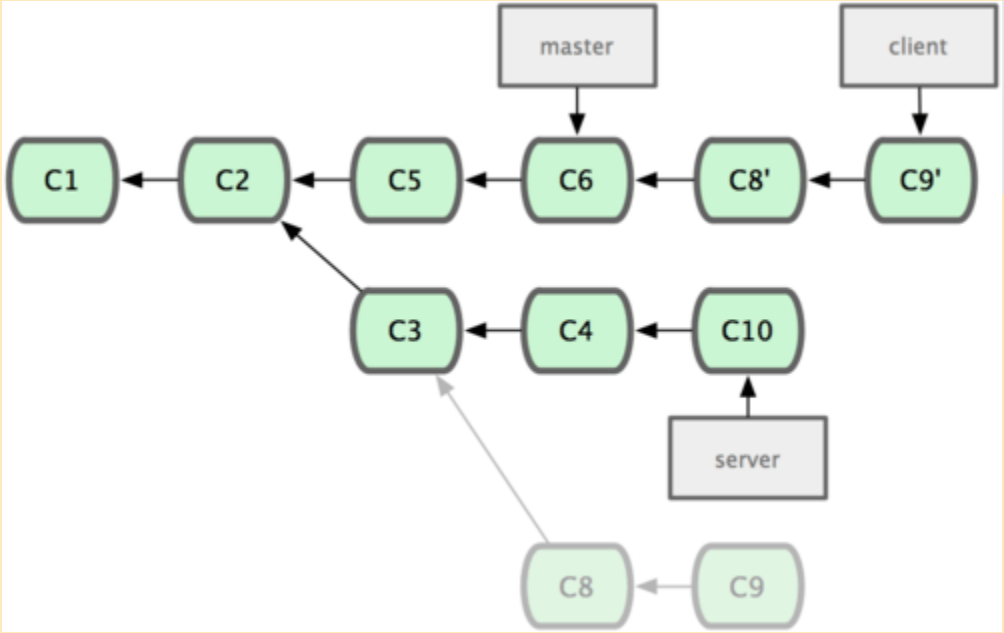
Usage

- first argument is the branch you rebase onto
- the second is the branch you are rebasing

Rebase Diagrams







Nice Video for Merge and Rebase

https://www.youtube.com/watch?v=Ypi4Kwx0GJw

Useful Rebase Options

- --onto moves current branch to branch off of the argument after onto
  - does not have to be a descendant
- --continue finishes a rebase after resolving merge conflicts
- --abort aborts current rebase
- -i interactive (shown in detail in lecture 3)

git rebase vs git merge

Rebase

- linearizes history
- keeps commits clear

Merge

- does not modify history

Git pull

- git pull = git fetch && git merge
- configure git pull to rebase automatically
- `git config branch.autosetuprebase always`
- `git config branch.*branch-name*.rebase true`

Git Workflows

- Many different ways to use git in a team
- There are general guidelines that people follow that allow for easy debugging, working efficiency, quality assurance, continual testing

Workflows Scenarios

- Working alone
- Working on a small team of trusted members
- Working on a large team where stability and availability are crucial
- Working on a project with untrusted contributors

Working Alone

- It is up to you what workflow model to use
- feature-based Branch model still the best
  - create branches off of master, do development, rebase them back into master
- Pull requests not necessary
- More freedom with rewriting history (can squash and rebase whenever you want)
- It is easy to keep a linear history

Working with a small, trusted team

- Everyone has write access
- pull requests not necessary
- use branching model
- “private” and “public” branches
- public branches examples are master and release branches, all commits should be clear and meaningful
- private/local branches can be thought of as scratch paper. Being local affords you the advantage of being able to squash and edit commit history.

Working on a large team with stability and availability

- Will utilize branching or forking using a pull request model
- Developers will have their own branches to do development and have to open pull requests to merge into master.
- Often utilizes Continuous Integration to prevent bad commits from being accepting into master
- use rebase to keep master linear and to keep his history intact
- rebase your topic branch onto origin/master
- pull request topic branch into master

Working with unknown, untrusted contributors

- Used in open source projects where the quality of contributors is not always known
- Git’s permission levels are very binary in that you either have write access to all the branches or to none of them
- A contributor must fork to their own repo,

git rm

- removes a file from the branch and from the index but does not modify working directory.
- --cached keeps it in your working directory
- only works if file is identical to branch tip

Next Week in Git Stuco

- Midterm

HW

Short HW on rebasing; released on friday

Due next week by friday 11:59PM

Midterm

- Midterm will cover everything so far, including basic material relating to rebase
- It will focus a lot on concepts we have learned, not just on the commands.
- Format: multiple-choice + short answer

List of Commands Covered in Detail

- git clone
- git add
- git commit
- git push
- git status
- git log

Continued

- git checkout
- git reset
- git merge
- git branch
- git pull
- git remote
- git init

More Command with Less Detail

- git rm
- git stash
- git clean
- git gc

Continued

- git tag
- git blame
- git diff

Important High Level

- Point of VCS
- Advantages of distributed/centralized
- Point of branching
- Difference between rebase and merge
- Merge conflicts
- Be able to define index, repository, branch, head, master, remote, origin, and working directory

Continued

- Feature based commits
- Hashes
- Change Sets vs Snapshots
- Function of .gitignore
- Difference between checkout and reset