Lecture 4 - Introduction to Version Control

DSE 511

Drew Schmidt
2022-09-06

Announcements

- Nothing unresolved from last time
- Questions?

Course Structure

- Introductory lecture
- Getting into the details
- Homework assigned

Module 1: Version Control

- Introduction to Version Control (9/6)
- Basic git (9/8)
- Working with Remotes (e.g. GitHub) (9/13)
- Collaborating with Others (9/15)
- When Things Go Wrong (9/20)
- Homework (~9/22)

What Is Version Control?

Many Names

- version control
  - vcs (version control system)
- source control
- revision control
- source management
- …

But Why Though?

Without version control

- my_project.code
- my_project2.code
- my_project_final.code
- my_project_final2.code
- my_project_final_FINAL.code
- my_project_final_FINAL2.code
- …

With version control

- You only operate on my_project.code
- Let vcs handle changes

But Why Though?

Without version control

```
# I MIGHT NEED THIS AGAIN SOME DAY
# f = function(...) {
#   do_this()
#   do_that()
#   etc()
# }
g = = function(...) {
  # ...
}
```

With version control

```
g = = function(...) {
  # ...
}
```

But Why Though?

Standardizes good practices

- Forward tracking changes
- Reverting a bad feature
- Backups

Implementations

- Monolithic - single codebase (commit is equivalent to commit + push)
  - svn
- Distributed - multiple codebases (commit separate from push)
  - git
  - mercurial (hg)
- There may be others; they're probably irrelevant

Implementations

- svn is very dated; I really don't recommend it
- git and hg are very similar
- We will only focus on git

What Is git?

- Version control software
- Command line tool
- Originally created by Linus Torvalds

What Is git?

> I'm an egotistical bastard, and I name all my projects after myself. First Linux, now git.

git

Why git?

- It's verbose
- It's basically the standard
  - Lots of references
  - Other people generally know it
- GitHub is great!
- Easy to use 95% of the time

Why Not git?

- It's verbose
- Multiple ways to do many things
- That last 5% is a doozy

File Tracking

Do Track

- Inputs
  - Text files
  - Scripts
  - Images
  - ...

Don't Track

- Outputs
  - Binary files
  - Generated files
  - knitr cache
  - ...

Some Terms

- **repo** - the code repository
- **remote** - your repo hosted somewhere else (e.g. GitHub)
- **clone** - creating a local copy of a repo
- **forking** - creating a remote copy of a repo
- **branch** - like a local fork

Collaborating

- Many remotes
  - GitHub
  - Bitbucket
  - GitLab
  - ...
- Many advantages
  - enables collaboration
  - free backups!
  - graphical interface to git's worst parts

Collaborating

- Using remotes (e.g. GitHub)
- Usual pipeline (distributed model)
  1. fork
  2. make changes
  3. create pull request (PR)
- For centralized codebases (e.g. proprietary ones), forking may be blocked
  1. branch
  2. make changes
  3. create PR

Using git

- Most people don't just use git
  - git + GitHub
  - git + Bitbucket
  - etc
- Handful of commands are most of what you'll ever need
  - `init`
  - `status`
  - `add`
  - `commit`
  - `push/pull`
- The web interface handles most of what else you will need

When Things Go Wrong

- git does what you tell it to do
  - This includes stupid things
- It's hard to *truly* wreck a git repo
  - But it's easy to wreck it beyond *your* ability to fix
- This can be very hard
- We have a single lecture dedicated just to this

Where To Get Help

- git documentation https://git-scm.com/doc
- Tutorials
  - https://www.atlassian.com/git
  - https://www.w3schools.com/git
  - https://www.tutorialspoint.com/git/index.htm
- Stack Overflow https://stackoverflow.com/questions/tagged/git

Wrapup

Wrapup

- Version control is a very useful tool
  - Change tracking
  - Backups/reverts
  - Collaboration
- git and GitHub are the standards
- Easy to get started
  - A handful of commands will take you far

Questions?