Recitation 6: Git and Github

Eugene Tan (TA)

10/16/2020

Outline for today

- Version Control
- Git commits as snapshots
- A sequence of commits as version history
- Repositories
- Collaborating on github
- Troubleshooting - commits at the same time
  - merging
  - rebasing

Version Control

⊞ lac_redraft_ET4.docx	⊘	Jul 6, 2019 at 4:32 PM	18.3 MB
⊞ lac_redraft_ET5.docx	⊘	Aug 8, 2019 at 5:56 PM	21.5 MB
⊞ lac_writeup_AJ.docx	⊘	Jun 10, 2019 at 7:53 AM	13.1 MB
⊞ lac_writeup_ET2.docx	⊘	Jun 17, 2019 at 5:38 PM	13.2 MB
⊞ lac_writeup_ET3.docx	⊘	Jun 24, 2019 at 5:36 PM	13.1 MB
⊞ lac_writeup.docx	⊘	Apr 19, 2019 at 5:19 PM	324 KB
⊞ Unbrekable_ARG_zero_draft_ET.docx	⊘	May 16, 2019 at 7:10 PM	312 KB
⊞ Unbrekable_ARG_zero_draft.docx	⊘	May 16, 2019 at 12:49 PM	147 KB

▶ Emailing your file back and forth
▶ Working over zoom
▶ Time difference

Version Control

Git solves this problem for code. Code is different from normal writing because your end depends on your beginning. Uses multiple files.

From datacamp:

- ▶ Nothing that is saved to Git is ever lost, so you can always go back to see which results were generated by which versions of your programs.
- ▶ Git automatically notifies you when your work conflicts with someone else's, so it's harder (but not impossible) to accidentally overwrite work.
- ▶ Git can synchronize work done by different people on different machines, so it scales as your team does.

Version Control - production and accountability

Nothing that is saved to Git is ever lost, so you can always go back to see which results were generated by which versions of your programs.

Version Control - preventing bugs in collaboration

Git automatically notifies you when your work conflicts with someone else's, so it's harder (but not impossible) to accidentally overwrite work.

Version Control - encourage collaboration on code

Git can synchronize work done by different people on different machines, so it scales as your team does.

Version Control - preventing bugs in collaboration

What my git repo looks like for this class

The *repository* is the folder in which everything is tracked A *commit* is a snapshot Public repo Private repo

Github

Github is just a place to store your git repos online. Allows collaboration.

Baby command-line tools

On git bash (windows) or on terminal (mac)
`computer_name:folder username$`

- ► ls (-a, -l)
- ► pwd
- ► cd (., .., ~, -)
- ► mkdir/rmdir
- ► open (-a)
- ► tab-autocomplete
- ► man
- ► control-a control-e

A commit is a snapshot.

`git add` to stage a commit. `git commit` to commit the staged items.
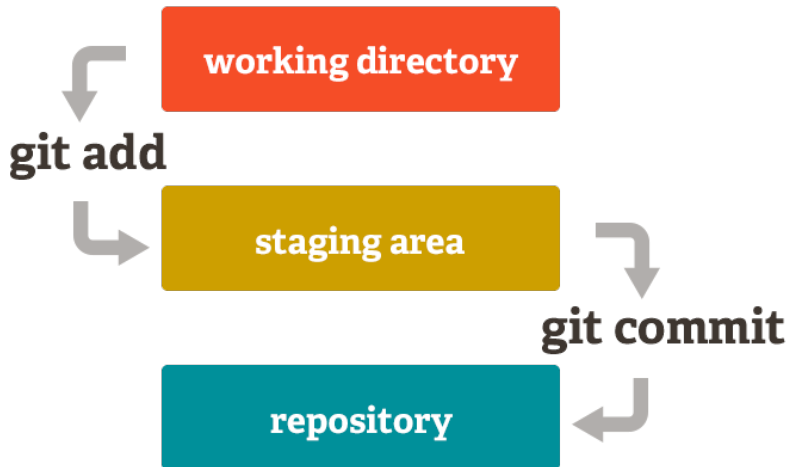
Demonstration.

Edit this file in this folder.

- ▶ `git status` checks the current status - relative to the last commit, what has changed, what has been staged, what is untracked?
- ▶ `git log` checks the version history through time (if you added the aliases for lg1 and lg2, `git lg1` does the same thing)
- ▶ `git diff` checks the differences between my current working directory and the last commit
- ▶ `git diff hash_before hash_after` checks the differences between two commits
- ▶ `git add filename` stages an edited file
- ▶ `git commit -m "my commit message` commits with the commit message 'my commit message'
- ▶ `git push` pushes to github.

Diff

What is the difference between the status now and the last commit?

Basic workflow

- ▶ Configure your name and email
- ▶ Configure security
- ▶ Create a git repository `git init` or `git clone`
- ▶ After you've made changes
  - ▶ Stage your changes `git add`
  - ▶ Commit a change `git commit`
  - ▶ Push your commits to GitHub `git push`
- ▶ After someone in your team has made changes
  - ▶ Get their commits from github `git pull`
  - ▶ Check what the differences they made are `git diff`
  - ▶ Work on it yourself

Configure your git identity

Set your username $ `git config --global user.name "John Doe"`

Set your email $ `git config --global user.email johndoe@example.com`

Configure others $ `git config --global init.defaultBranch main`

Get a list of your current setup $ `git config --list`

Some extra shortcuts to show the pretty logs I use above.

Make an alias for git lg1 (from here) $ git config --global alias.lg1 "log --graph --abbrev-commit --decorate --format=format:'%C(bold blue)%h%C(reset) - %C(bold green)(%ar)%C(reset) %C(white)%s%C(reset) %C(dim white)- %an%C(reset)%C(bold yellow)%d%C(reset)' --all"

Make an alias for git lg2 $ git config --global alias.lg2 "log --graph --abbrev-commit --decorate --format=format:'%C(bold blue)%h%C(reset) - %C(bold cyan)%aD%C(reset) %C(bold green)(%ar)%C(reset)%C(bold yellow)%d%C(reset)%n''   %C(white)%s%C(reset) %C(dim white)- %an%C(reset)' --all"

Cryptography (Setup is advanced)

If you don't want to enter your github password every time, then set a key using this link

- ▶ One-way security
- ▶ You hold a private key that is linked to a public key

Secured Socket Layer (SSH) Process

SSH Client

SSH Server

User Connect to Server

Server Sends Public Key

Parameters Are Established and Secured Channel in Opened

User Login to Server Host Operating System

SECRET
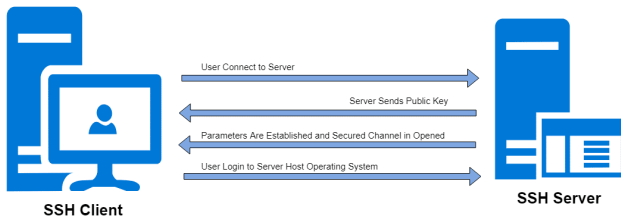
Demonstration

- ▶ Volunteer will make a private repository on github called 'ee2'.
- ▶ Share it with me and her group member.
- ▶ Clone it to her folder for this class.
- ▶ Adds the first part of her homework ee2. Commit and push.
- ▶ I will clone it.
- ▶ Make an edit, add, commit, push.
- ▶ Liz will pull the changes, diff to see what I have done, and then continue working.

Now we'll cause an error

What happens if we make a change at the same time?

- ▶ She will make an edit to the file and commit
- ▶ She will push
- ▶ I will make an edit to the file and commit
- ▶ I will try to push

What is the problem here?

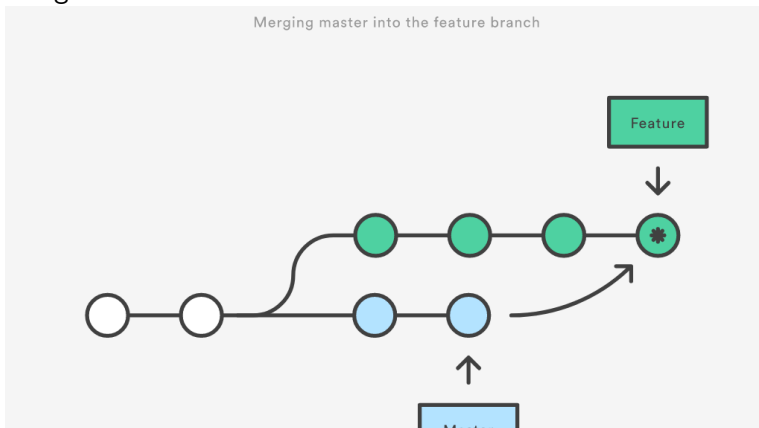*I need to pull before I edit the file. Editing the file at the same time causes conflicts.*

Dealing with conflicts

Easy way

▶ look at my file and her file and manually sort out the diferences, commit that
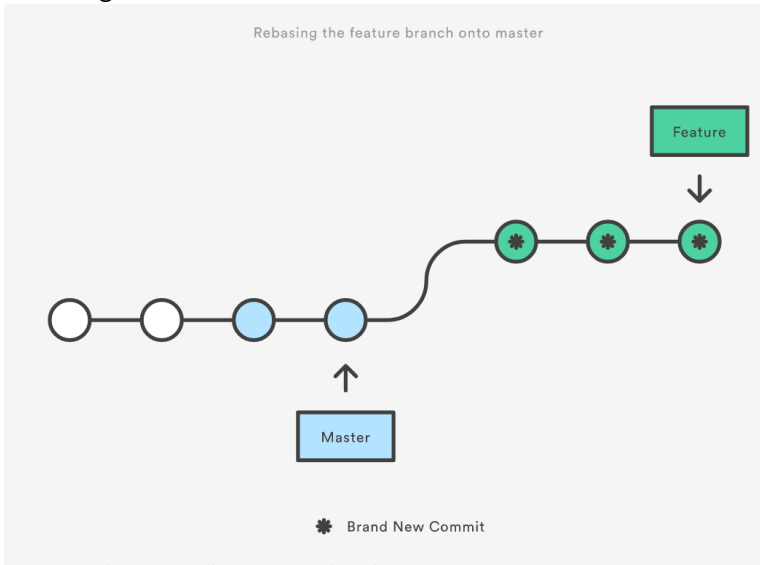
Intermediate way

▶ Merge branches

Merging master into the feature branch

Advanced way

- ► Rebasing

Rebasing the feature branch onto master

Feature

Master

❋ Brand New Commit

Branches (Intermediate)

Tips

- Set up your repo on github.
- share your repo with your partner before you add anything
- commit whenever you paste something in,
- commit before and after you get each thing working
- commit often
- push often
- write informative commit messages