Exercises

Clone the repository for the class website.

```
$ git clone https://github.com/missing-semester/missing-semester.git
Cloning into 'missing-semester'...
remote: Enumerating objects: 1928, done.
remote: Counting objects: 100% (30/30), done.
remote: Compressing objects: 100% (21/21), done.
remote: Total 1928 (delta 11), reused 21 (delta 7), pack-reused 1898
Receiving objects: 100% (1928/1928), 15.54 MiB | 5.12 MiB/s, done.
Resolving deltas: 100% (1137/1137), done.
$ cd missing-semester
```

Explore the version history by visualizing it as a graph.

```
missing-semester$ git help log
missing-semester$ git log --all --graph --decorate
* commit c0c78435629f4ecfaaaa1bb854b19bfaaa9f3b48 (HEAD -> master,
origin/master, origin/HEAD)
| Author: Anish Athalye <me@anishathalye.com>
| Date:   Sun May 2 15:42:07 2021 -0400
|
|     Update link
|
|     This patch also makes a flaky URL be ignored by CI.
|
*   commit 91fb22c0582f1b84e2ab10a6534ccdca0e1d483b
...
```

Who was the last person to modify `README.md`? (Hint: use `git log` with an argument).

```
# View latest commit history of README.md
$ git log -1 README.md
commit 79d143ff650d59a2d69961b2c68d697096c0e8f4
Author: Anish Athalye <me@anishathalye.com>
Date:   Sat Dec 26 09:42:23 2020 -0500

    Switch to proof-html action
# Show only the author’s name
$ git log -1 --format=%an README.md
Anish Athalye
```

What was the commit message associated with the last modification to the `collections:` line of `_config.yml`? (Hint: use `git blame` and `git show`).

```
# Find out at which commits lines of _config.yml was modified
$ git blame _config.yml
...
^112ddbd (Anish Athalye 2019-01-04 22:00:31 -0500 17)
a88b4eac (Anish Athalye 2020-01-17 15:26:30 -0500 18) collections:
a88b4eac (Anish Athalye 2020-01-17 15:26:30 -0500 19)   '2019':
a88b4eac (Anish Athalye 2020-01-17 15:26:30 -0500 20)     output:
true
a88b4eac (Anish Athalye 2020-01-17 15:26:30 -0500 21)   '2020':
/collections
# Show log of commit a88b4eac
$ git show a88b4eac
commit a88b4eac326483e29bdac5ee0a39b180948ae7fc
Author: Anish Athalye <me@anishathalye.com>
Date:   Fri Jan 17 15:26:30 2020 -0500

    Redo lectures as a collection
...
$ git help show
# Show only the commit message
$ git show -s --format=%B -n 1 a88b4eac
Redo lectures as a collection
```

One common mistake when learning Git is to commit large files that should not be managed by Git or adding sensitive information. Try adding a file to a repository, making some commits and then deleting that file from history (you may want to look at this).

```
# Being careless with git add without checking git status
password-protection-program$ git add .
$ git commit -m "Added export functionality"
$ git log --oneline
c7973e3 (HEAD -> master) Added export functionality
7dc2dcd Added sample program
da1798f Added README
# Delete sensitive file from history
$ git filter-branch --force --index-filter "git rm --cached
--ignore-unmatch book.xlsx" \
> --prune-empty --tag-name-filter cat -- --all
Proceeding with filter-branch...

Rewrite da1798fc64ce3789c386c9ae65746d3e15b00294 (1/3) (0 seconds
passed, rRewrite 7dc2dcd5f5904d025076675b2840627f8732dd73 (2/3) (0
seconds passed, rRewrite c7973e3e6faf57b6fa6dbdb441ad87e142834dc5
(3/3) (0 seconds passed, remaining 0 predicted)    rm 'book.xlsx'

Ref 'refs/heads/master' was rewritten
# Show commit logs
$ git log --oneline
7dc2dcd (HEAD -> master) Added sample program
da1798f Added README
# Adding file to .gitignore for extra caution
$ echo 'book.xlsx' >> .gitignore
```

Clone some repository from GitHub, and modify one of its existing files. What happens when you do `git stash`? What do you see when running `git log --all --oneline`? Run `git stash pop` to undo what you did with `git stash`. In what scenario might this be useful?

```
$ git clone https://github.com/anishathalye/dotfiles.git
$ vim bashrc
# Functions
```

```
source ~/.shell/functions.sh

# Will I still be here after git stash?

# Allow local customizations in the ~/.shell_local_before file
...
# Running git stash reverted the directory to the HEAD commit
$ git stash
Saved working directory and index state WIP on master: 557f082 Update dates
# The modifications I made was stashed away and logged
$ git log --all --oneline
230df68 (refs/stash) WIP on master: 557f082 Update dates
da5a520 index on master: 557f082 Update dates
557f082 (HEAD -> master, origin/master, origin/HEAD) Update dates
# Running git stash pop brought back the changes I made
$ git stash pop
On branch master
Your branch is up to date with 'origin/master'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
	modified:   bashrc

no changes added to commit (use "git add" and/or "git commit -a")
Dropped refs/stash@{0} (230df68436050ad6854aaa4a675b0a701b10b357)
```

When I ran `git stash`, the changes I made on the bashrc file were undone. My working directory was brought back to before I made the change. Running `git log --all --oneline` showed me that my changes weren't gone but was stashed away. `git stash` seems handy when I'm not quite ready to commit my changes and work on something else in the meantime. In that way, I could have a sort of draft commit saved for later while working on new commits.

Like many command line tools, Git provides a configuration file (or dotfile) called `~/.gitconfig`. Create an alias in `~/.gitconfig` so that when you run `git graph`, you get the output of `git log --all --graph --decorate --oneline`.

```
$ vim ~/.gitconfig
[alias]
        graph = log --all --graph --decorate --oneline
$ git help graph
'graph' is aliased to 'log --all --graph --decorate --oneline'
```

You can define global ignore patterns in `~/.gitignore_global` after running `git config --global core.excludesfile ~/.gitignore_global`. Do this, and set up your global gitignore file to ignore OS-specific or editor-specific temporary files, like `.DS_Store`.

```
$ git config --global core.excludesfile ~/.gitignore_global
$ vim ~/.gitignore_global
# OS-generated files
.Trashes
Thumbs.db
...
# Temporary files
*.tmp
*.bak
# Compiled source
# Packages
$ ls
 PassProtect.class  'Project Proposal - Password Protection
Program.docx'
 PassProtect.java    README.md
$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
      Project Proposal - Password Protection Program.docx
```

Fork the repository for the class website, find a typo or some other improvement you can make, and submit a pull request on GitHub.

```
$ git clone https://github.com/rareloto/missing-semester.git
# Track original repository for future pull requests
$ git remote add --track master upstream
https://github.com/missing-semester/missing-semester.git
$ git fetch upstream
From https://github.com/missing-semester/missing-semester
 * [new branch]      master     -> upstream/master
# Creating new branch for adding improvement
$ git checkout -b update-commandline-rephrase upstream/master
Branch 'update-commandline-rephrase' set up to track remote branch
'master' from 'upstream'.
Switched to a new branch 'update-commandline-rephrase'
# Making modifications on command-line.md
$ vim _2020/command-line.md
...
$ git add _2020/command-line.md
$ git status
On branch update-commandline-rephrase
Your branch is up to date with 'upstream/master'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
	modified:   _2020/command-line.md

$ git commit -m "Minor rephrasing on Job control Exercise 2"
$ git push -u origin update-commandline-rephrase
remote: Create a pull request for 'update-commandline-rephrase' on
GitHub by visiting:
...
```