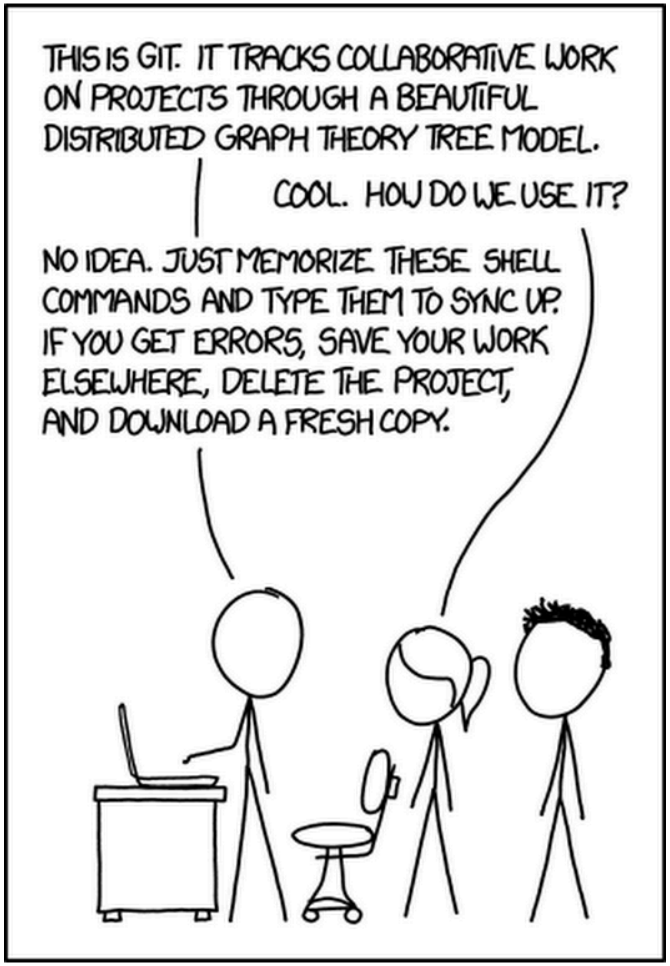
Git & GitHub


THIS IS GIT. IT TRACKS COLLABORATIVE WORK ON PROJECTS THROUGH A BEAUTIFUL DISTRIBUTED GRAPH THEORY TREE MODEL.
COOL. HOW DO WE USE IT?
NO IDEA. JUST MEMORIZE THESE SHELL COMMANDS AND TYPE THEM TO SYNC UP. IF YOU GET ERRORS, SAVE YOUR WORK ELSEWHERE, DELETE THE PROJECT, AND DOWNLOAD A FRESH COPY.

Git & GitHub

- **Git** is a system for version control
- **GitHub** is a web-based platform for hosting and sharing Git repositories
- Other version control software:
  - Apache Subversion (SVN)
  - Mercurial
  - Veracity

But GitHub has helped make Git very popular.

Git & GitHub

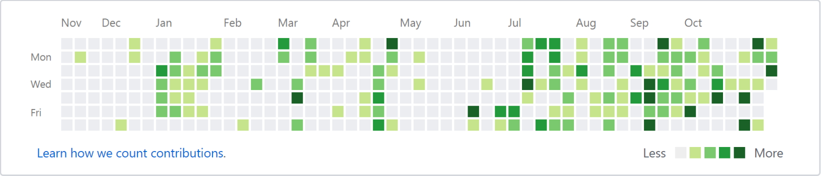
- Version control
  - – Keeps record of your changes
  - – Allows for collaborative development
  - – Keeps details record of *who* made *what* changes and *when*
  - – Allows you to revert to any and all changes **previously committed to** the repository

Git & GitHub

Typical workflow

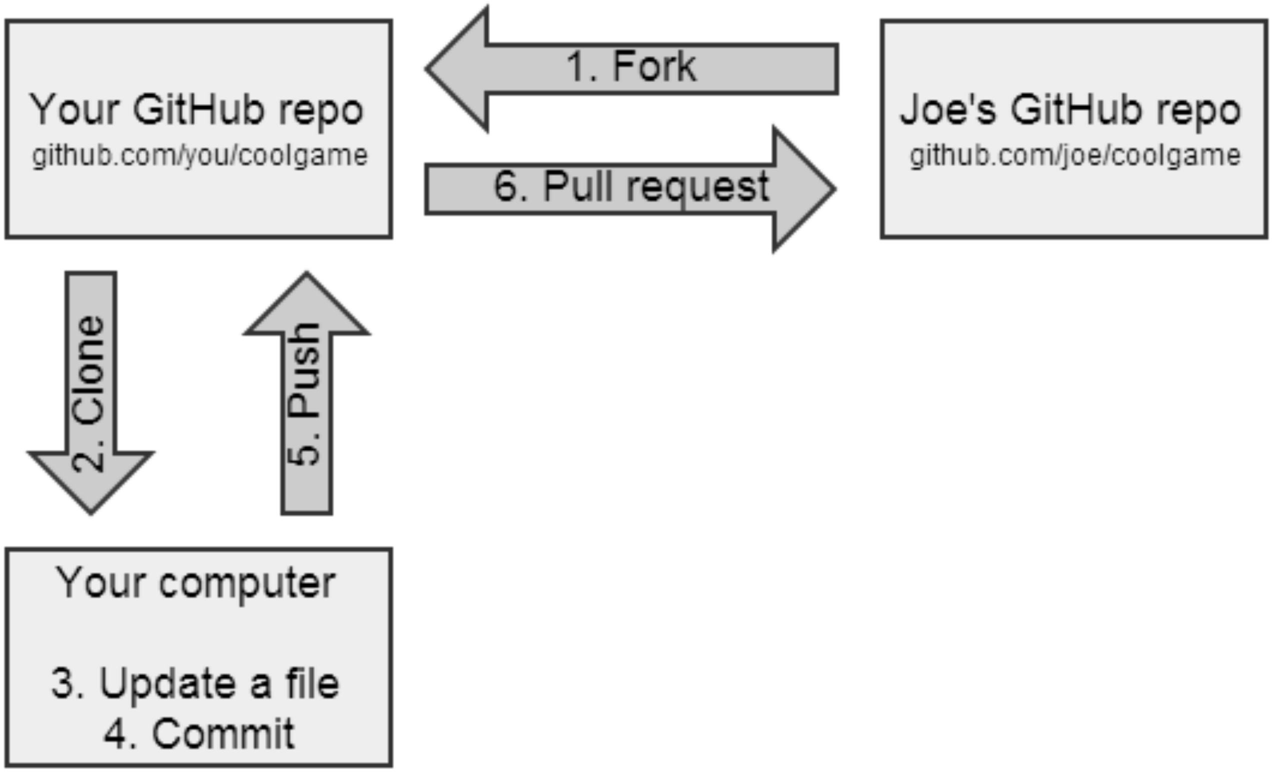
1. Clone a repository from GitHub (or create your own)
2. Change the code, add & commit your changes
3. Pull any changes from the master repository
4. Push your committed changes to the remote repo

- Repeat steps 2-4 for the rest of your life.



*This is an example with a *very* limited scope, but it is meant to illustrate one common practice

Git & GitHub



Your GitHub repo
github.com/you/coolgame

1. Fork

6. Pull request

Joe's GitHub repo
github.com/joe/coolgame

2. Clone

5. Push

Your computer

3. Update a file
4. Commit

http://www.dataschool.io/simple-guide-to-forks-in-github-and-git/

Git concepts

- Repository
- Commit
- Push
- Branch
- Pull
- Fork
- Merge

Git concepts

- Repository
  - Stores current and historical code, “repo” for short
  - Can be local or remote
- Commit
- Push
- Branch
- Pull
- Fork
- Merge

Git concepts

- Repository
- Commit
  - Used to save modified code to the repo
- Push
- Branch
- Pull
- Fork
- Merge

Git concepts

- Repository
- Commit
- Push
  - Sends committed changes to the remote repo
- Branch
- Pull
- Fork
- Merge

Git concepts

- Repository
- Commit
- Push
- Branch
  - Separate path for new code, can later be merged into main, “master,” branch
- Pull
- Fork
- Merge

Git concepts

- Repository
- Commit
- Push
- Branch
- Pull
  - Request to add modified code from one branch to another
- Fork
- Merge

Git concepts

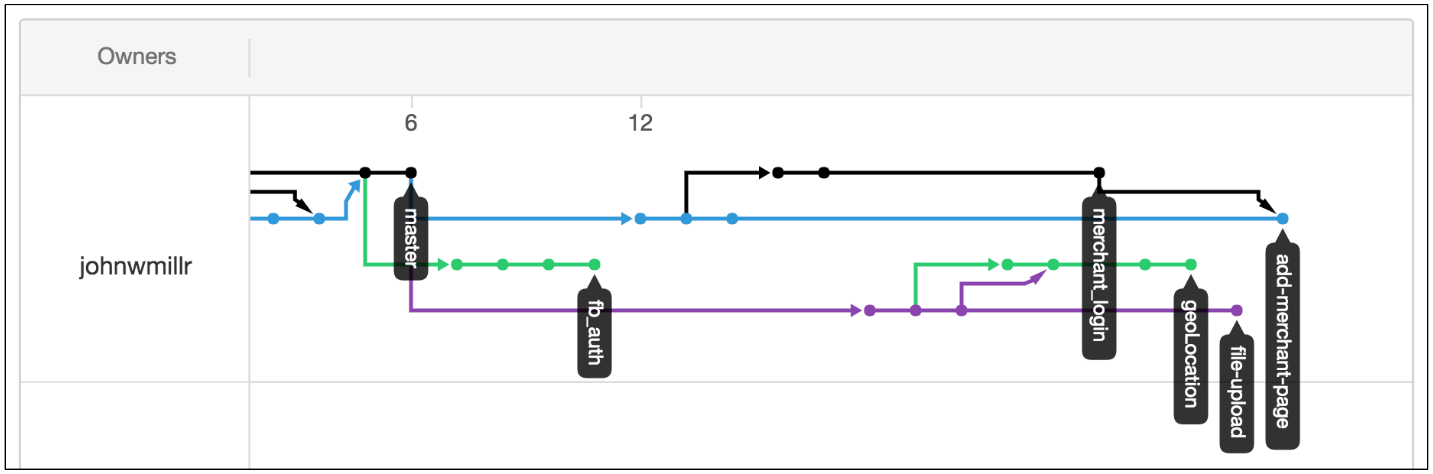
- Repository
- Commit
- Push
- Branch
- Pull
- Fork
  - Diverging copy of a repo, for new development
- Merge

Git concepts

- Repository
- Commit
- Push
- Branch
- Pull
- Fork
- Merge
  - Combine code from two branches

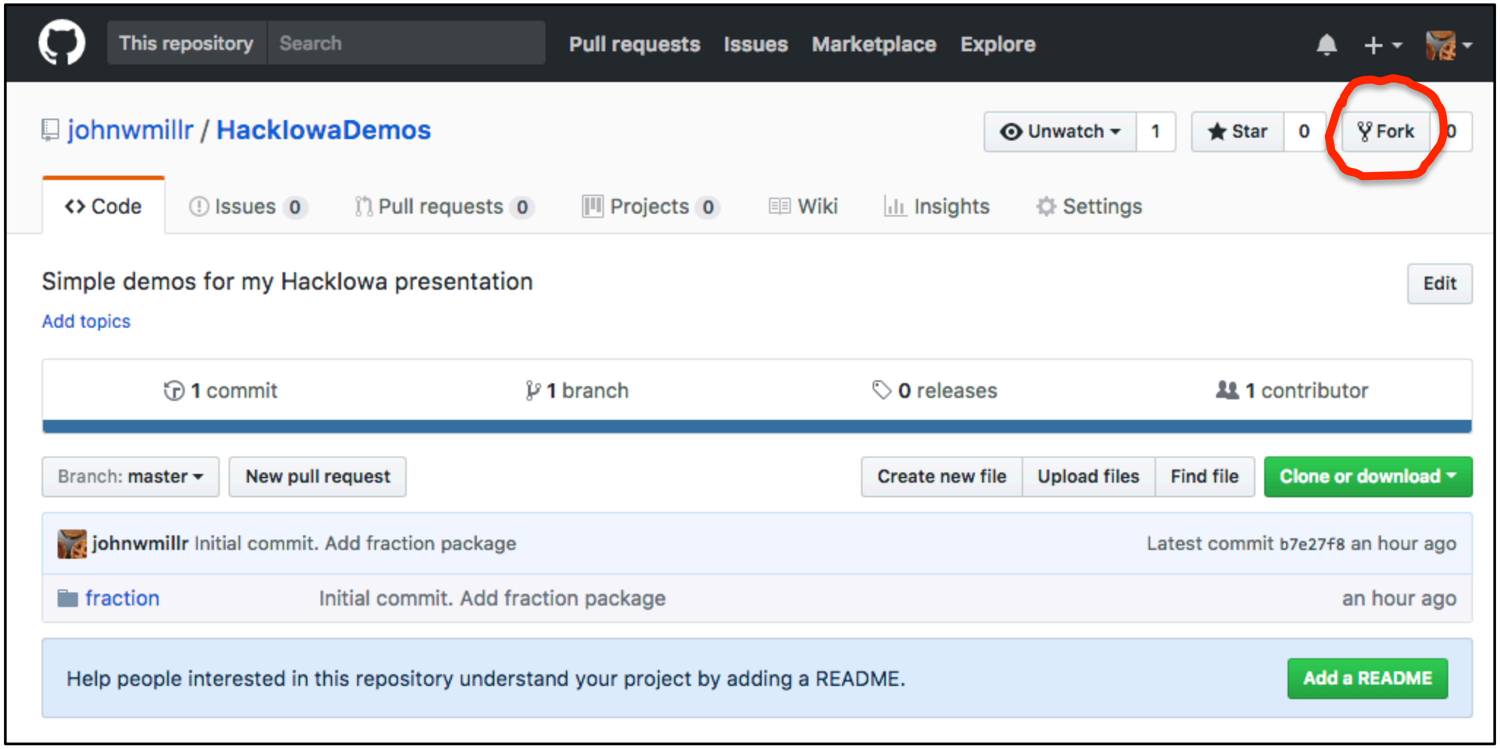
Git & GitHub

Project branches



GitHub Demo

1. Fork the repo
   - Link: https://github.com/johnwmillr/HackIowaDemos



Simple demos for my HackIowa presentation

johnwmillr Initial commit. Add fraction package — Latest commit b7e27f8 an hour ago

fraction — Initial commit. Add fraction package — an hour ago

Help people interested in this repository understand your project by adding a README.

GitHub Demo

2. Clone your forked repo

```
$git clone https://github.com/<your-name>/HackIowaDemos.git
$cd HackIowaDemos
$git status
```

3. Try out the fraction class

```
$cd fraction
$python
>>>from fraction import Fraction
>>>f1 = Fraction(1,2); print(f1)
>>>f2 = Fraction(1,6); print(f2)
>>>f1+f2
>>>f1.eval
```

GitHub Demo

4. Make a new branch

```
$git checkout –b fix-fraction-eval
```

5. Edit the code (make sure it works!)

6. **Add** and **commit** your changes

```
$git status
$git add .
$git status
$git commit –m “Fix the self.eval bug in fraction class”
```

7. Merge your modified branch into master

```
$git checkout master
$git merge fix-fraction-eval
$git log
```

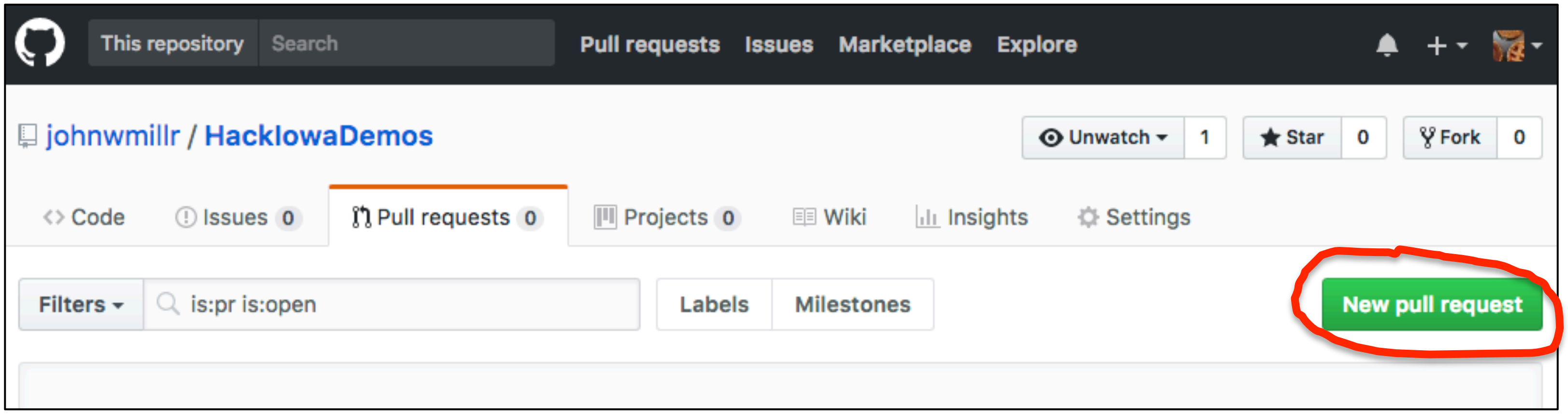
GitHub Demo

8. Push your changes to your remote repository

```
$git push origin master
$git status
```

9. View your changes on your GitHub repository
   - https://github.com/<your-name>/HackIowaDemos

10. Create a pull request into my repository
    - https://github.com/johnwmillr/HackIowaDemos/pulls



That’s it!

- Questions?