Part 4

Remotes and pull

Distributed version control system

Everone has their own complete copy of the repo

Github (or others) just a convenient place for sharing

For a sole developer

Github is

- showcase
- backup
- moving between machines

Github and keys: who can write to a repo?

git/ssh protocol

- crypto key identifies machines
- tell Github the keys
- works seamlessly
- effort to set up

https protocol

- no setup
- give password with every change

Github Desktop app

- no setup
- no password needed
- limited

Publishing with Github desktop

Connect to your account

Add a local repository

Publish

Look on Github

Danger: public commits shouldn't be changed

`git revert` "undoes" a commit

Creates a *new* commit that does the inverse changes

Concept: remote

Any number of remotes

Typically, just one, nicknamed `origin`

```
$ git show remote -v
```

Concept: remote branches

A branch on a remote

Called `<remote name>/<branch name>`

- `origin/master`
- `origin/capitals`

You can't move them

- `git branch ftr origin/feature` to create them
- `git fetch` updates the pointers
- `git branch -d origin/feature` deletes your pointer to it

Remote branches update as others change them

Concept: remote tracking branches

A *local* branch that's connected to a *remote* ("upstream") branch

`git fetch` will update remote pointers

`git pull` will `fetch` **and** `merge` changes into your local branch

`git push` will apply your local commits to that remote branch

`git push --set-upstream origin new_branch` pushes, and sets upstream

- ...if you authenticate on the command line

Update and push

- Create, checkout a new branch
- Create a new file on this branch
- `$ git log --oneline`
- Refresh `gitk`
- Push the changes (use Github Desktop)
- Refresh the repo on Github website
- `$ git log --oneline`
- Refresh `gitk`

Concept: Fork vs clone

Fork makes a copy of somone else's repo, places it in your account

- remembers where it came from
- Github extension (now others too)
- Not part of Git itself

Clone makes a copy of a repo, copies it to your PC

- keeps the source as the `origin` remote
- Don't need to fork first (but can't make changes to *that* repo)

Github collaboration models

Separate repos

- One "blessed repo"
    - Limited access
- Each person has own forked version
- Changes proposed by "pull requests"
- Maintainer adds the approved one

Single repo

- Everyone has access to same repo
- (optional) Merging to `master` branch limited
- Needs trust between collaborators

Propose a change to another's repo

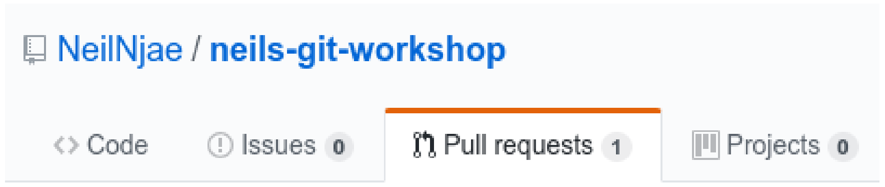
1. Talk to the person on your *left*
2. Find their repo and fork their repo (on Github website)
3. Clone the fork to your PC (use Github Desktop)
4. Create a new branch
5. Add a file saying something nice about the person to your left
    - Remember this is all public
6. Push the change (use Github Desktop)
7. Create a **pull request** (use Github website)

Handle the pull request

On Github website, go back to your own repo

Should see a *pull request*

Look at the file, make comments.



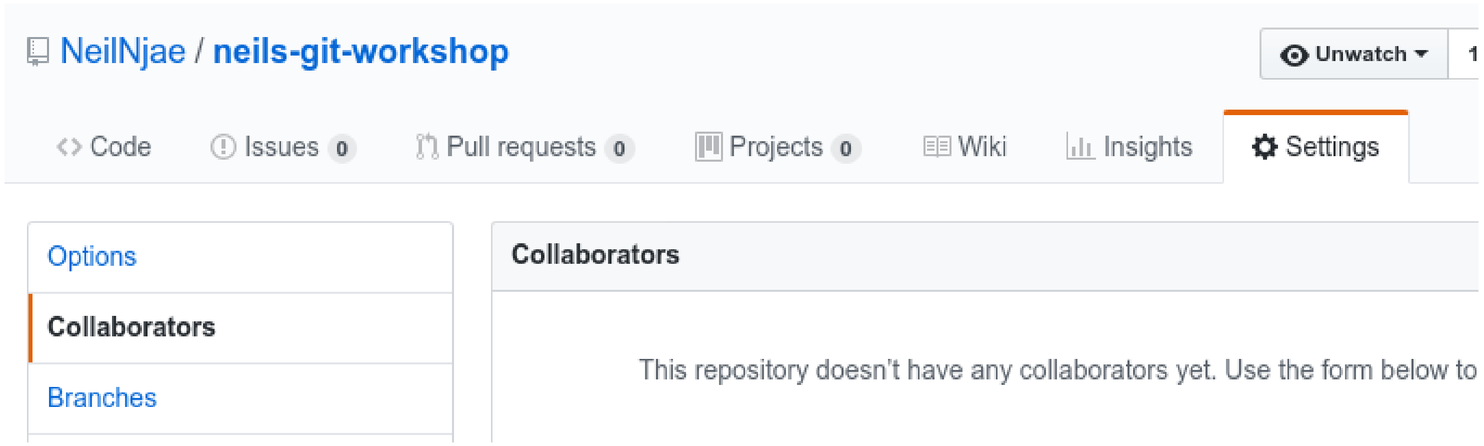
Merge the change into your repo

Delete the branch

Invite another to work on your repo

1. Talk to the person on your *right*
2. Ask nicely if they will add you as a collaborator

When you're asked, add the person


NeilNjae / neils-git-workshop

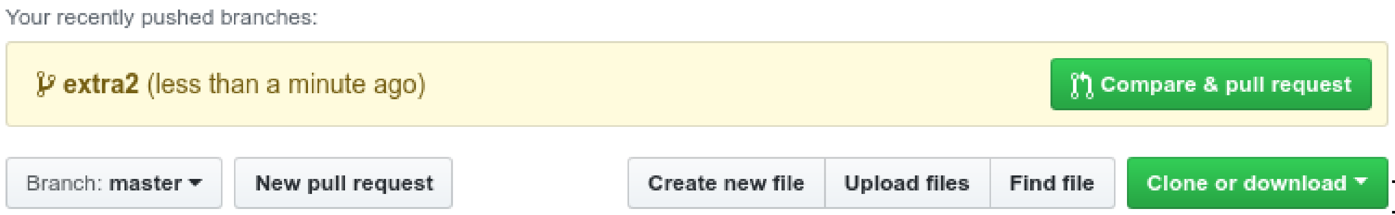
Collaborators

This repository doesn't have any collaborators yet. Use the form below to

Collaborating

Once you're a collaborator

1. Clone their repo (use Github desktop)
2. Create a new branch (however)
3. Add a file saying something nice about the person to your right
    - Remember this is all public
4. Publish or Push the change (use Github Desktop)
5. Open a **pull request** (use Github website)


Your recently pushed branches:

extra2 (less than a minute ago)

]

End of part 4

- local and remote
- fetch, pull, push
- fork and clone
- pull request

We are done!

Good resources

Git

- The Git book
- Official Git reference
- Atlassian Git tutorials, especially the advanced tutorials.
- Fixing mistakes with Git

Workflow

- Simple Git workflow from Atlassian
- Creative history rewriting