UNIVERSITY of WASHINGTON

Software Design for Data Science

Git, Part 2

Naomi Alterman
University of Washington
February 3, 2026

Review

How do I…

- view the change history for a repository?
  `git log`
- see what files in the directory are modified since the last commit?
  `git status`
- examine the changes between modified files and the last commit?
  `git diff`
- place a modified file into the staging area?
  `git add <file>`
- move the staged files into the repository history?
  `git commit -m “My commit message”`

Review

What do the following verbs do?

- Push
- Clone
- Pull

Review

What is…

- `HEAD`?
- `HEAD~3`?

Exercise

- Get into teams of 2-3
- Identify someone to “own” a repository who will create it in their GitHub account (PUBLIC repository)
- Add a README.md, LICENSE and .gitignore using the UI
  - Python language & MIT license are good to start
- Add the other team members as collaborators to the repository
- Everyone clone on their computer

Exercise

To add collaborators to a repository

- Go to the repo
- Click on Settings, then Collaborators & Teams, then Add People

You haven't added any teams or people yet

Organization owners can manage individual and team access to the organization's repositories. Team maintainers can also manage a team's repository access. Learn more about organization access

Add people | Add teams

Exercise

- Everyone make changes to the README.md
- Status, diff, add, commit, push
- What happens?

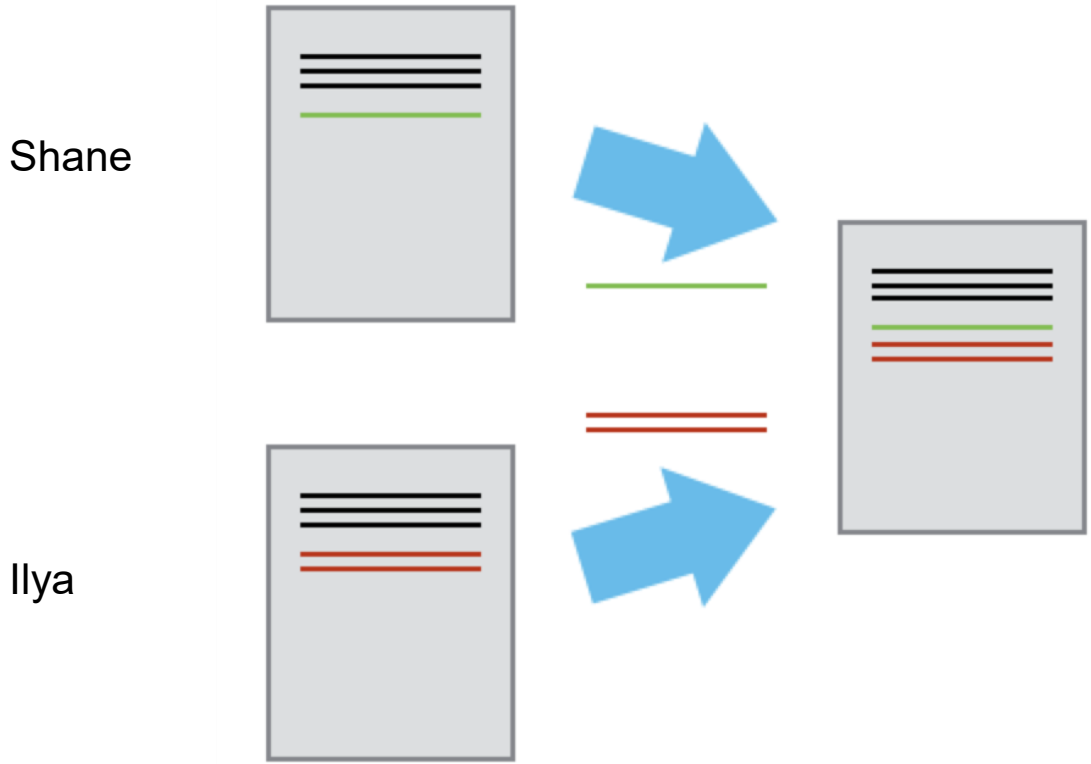


Figure: Multiple versions can be merged

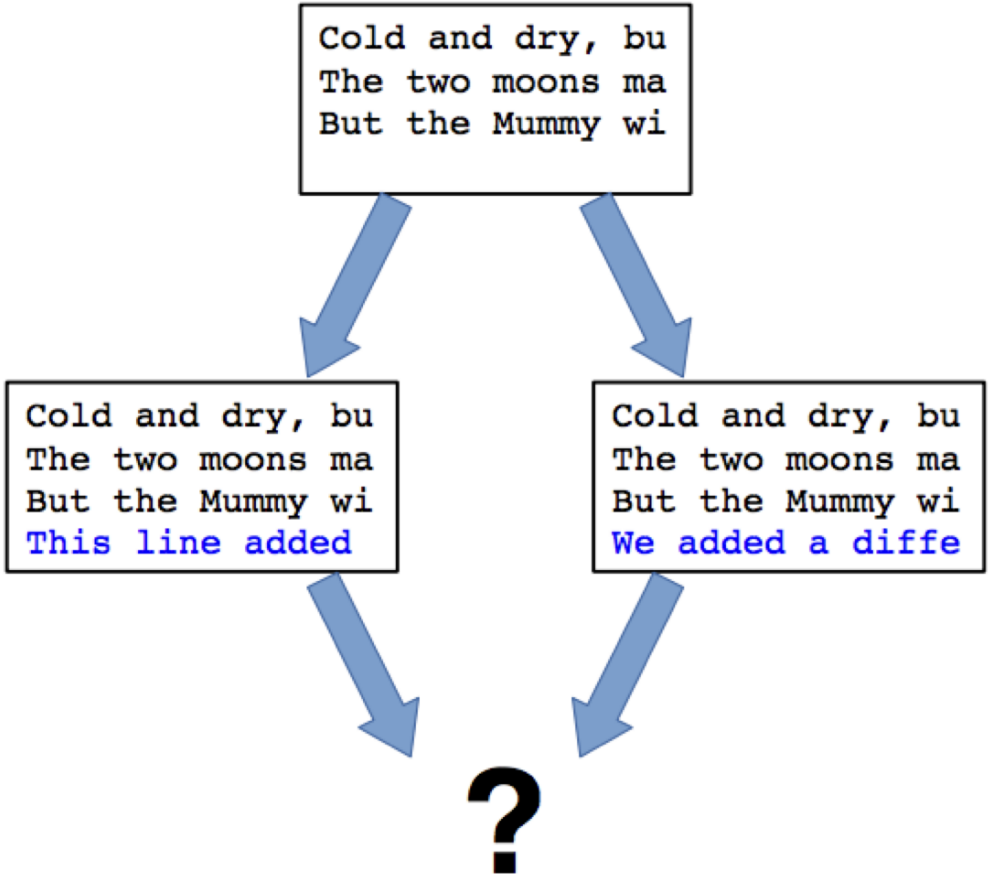


Figure: The conflicting changes

Merge conflicts

```
$ cat merge.txt
<<<<<<< HEAD
this is some content to mess with
content to append
=======
totally different content to merge later
>>>>>>> new_branch_to_merge_later
```

Fixing your merge conflict

```
git fetch
```

```
git merge origin
```

For each file with a reported conflict:

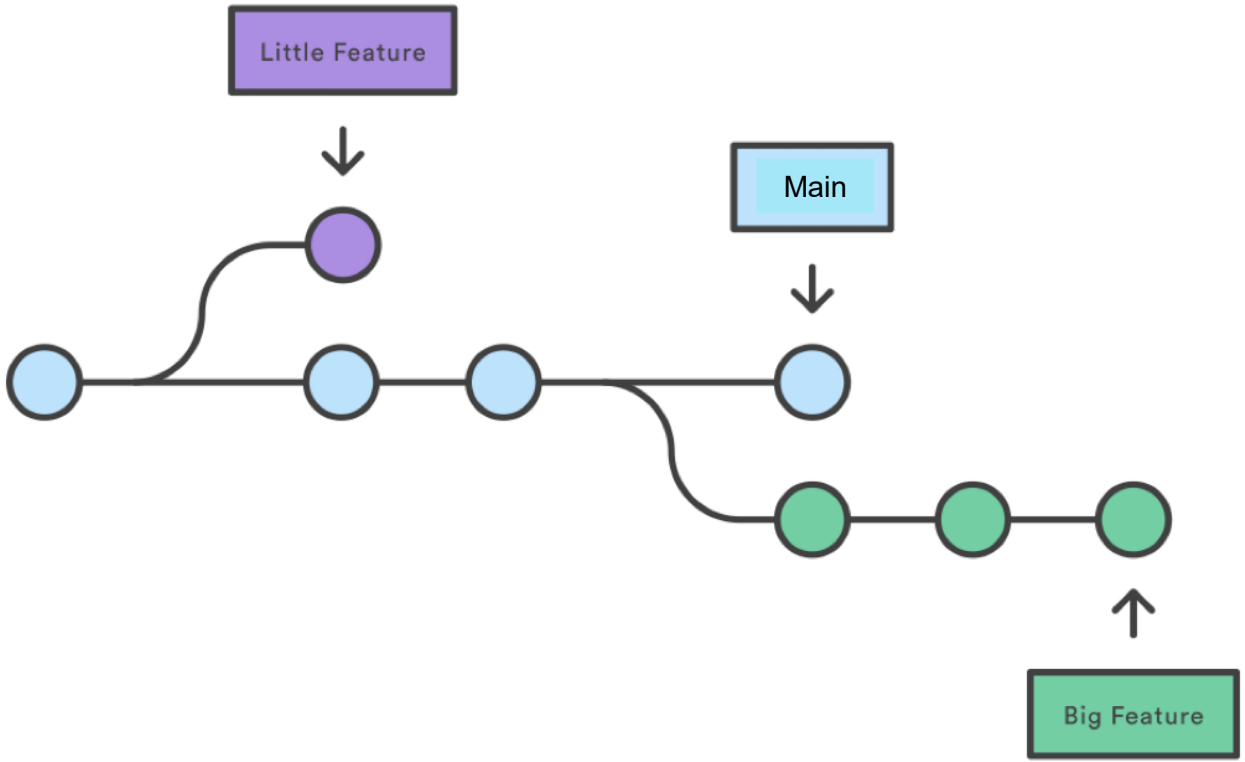
- Open the file in a text editor
- Find the conflicts, which start with lots of “<<<<<<“s and end with “>>>>>>”s
- Rewrite the block between the brackets to what you want it to be
- Remove the brackets and equals signs too
- `git add` the file

```
git commit
```

Collaborating Scenario

- Main development trunk of codebase
- Bug comes in via issue report on GitHub
- You need to work on the bug but don’t want to screw up the main development trunk
- What to do?

Branches



Branch commands

- Create a new branch on your local computer

  ```
  git branch name-of-branch
  ```

- Delete a branch

  ```
  git branch -D name-of-branch
  ```

- Switch to a branch (or main)

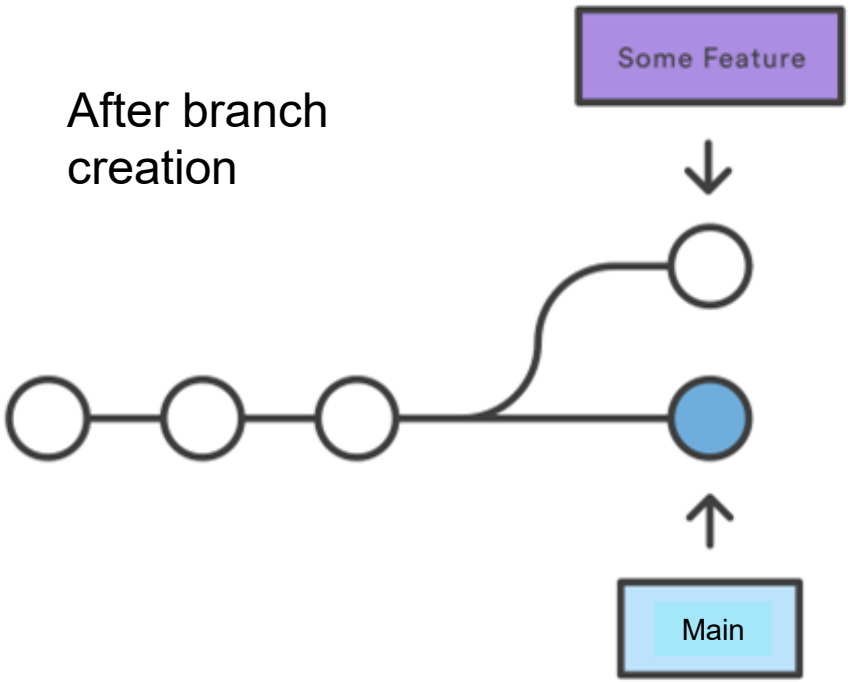
  ```
  git checkout name-of-branch
  git checkout main
  ```

- Make a new branch and switch into it all at once

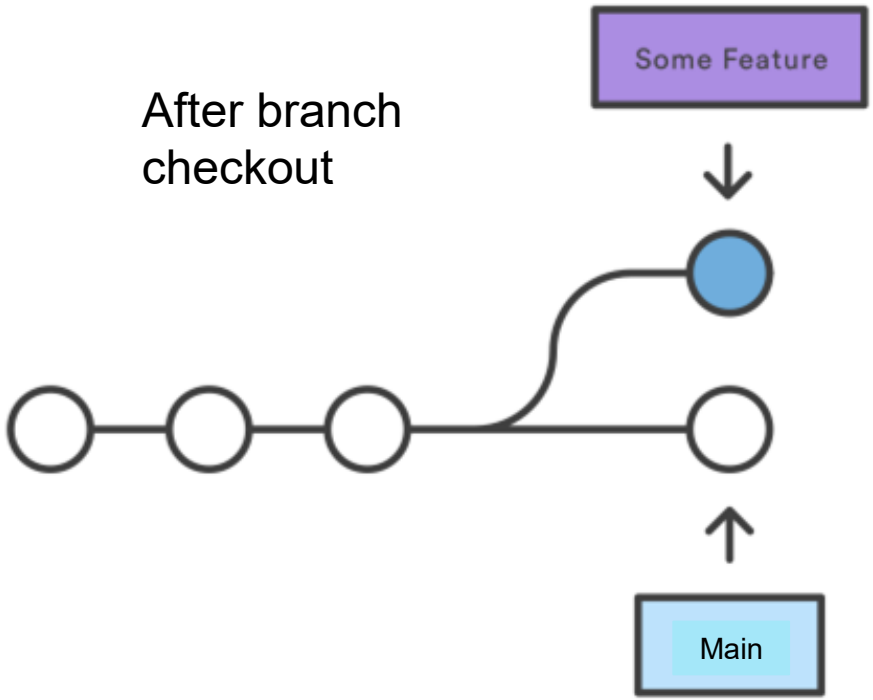
  ```
  git checkout -b name-of-new-branch
  ```

Branches: visual

After branch creation

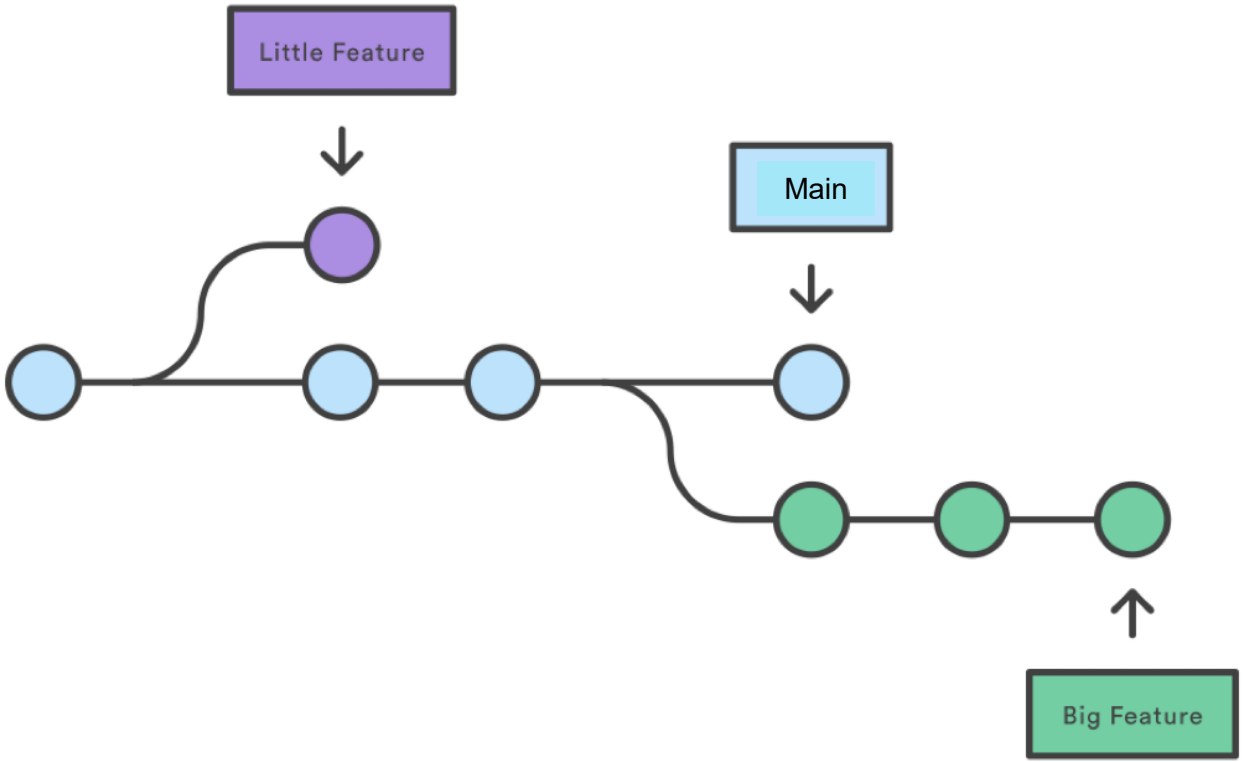


After branch checkout

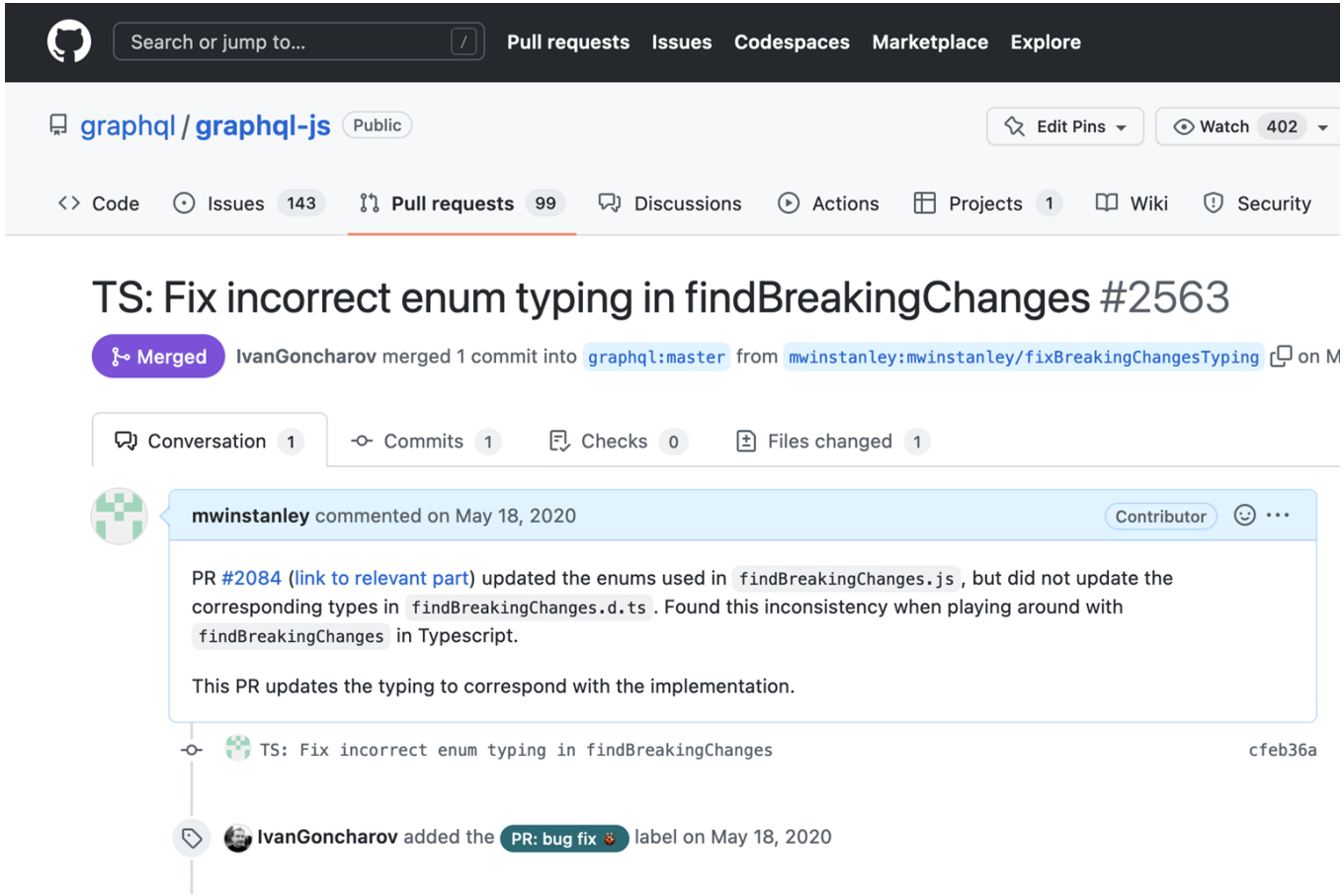


Merges

But how do we get our features back into main?



Pull requests



Submitting a pull request

From your branch, push the branch to GitHub:

```
git push
```

You'll need to configure git:

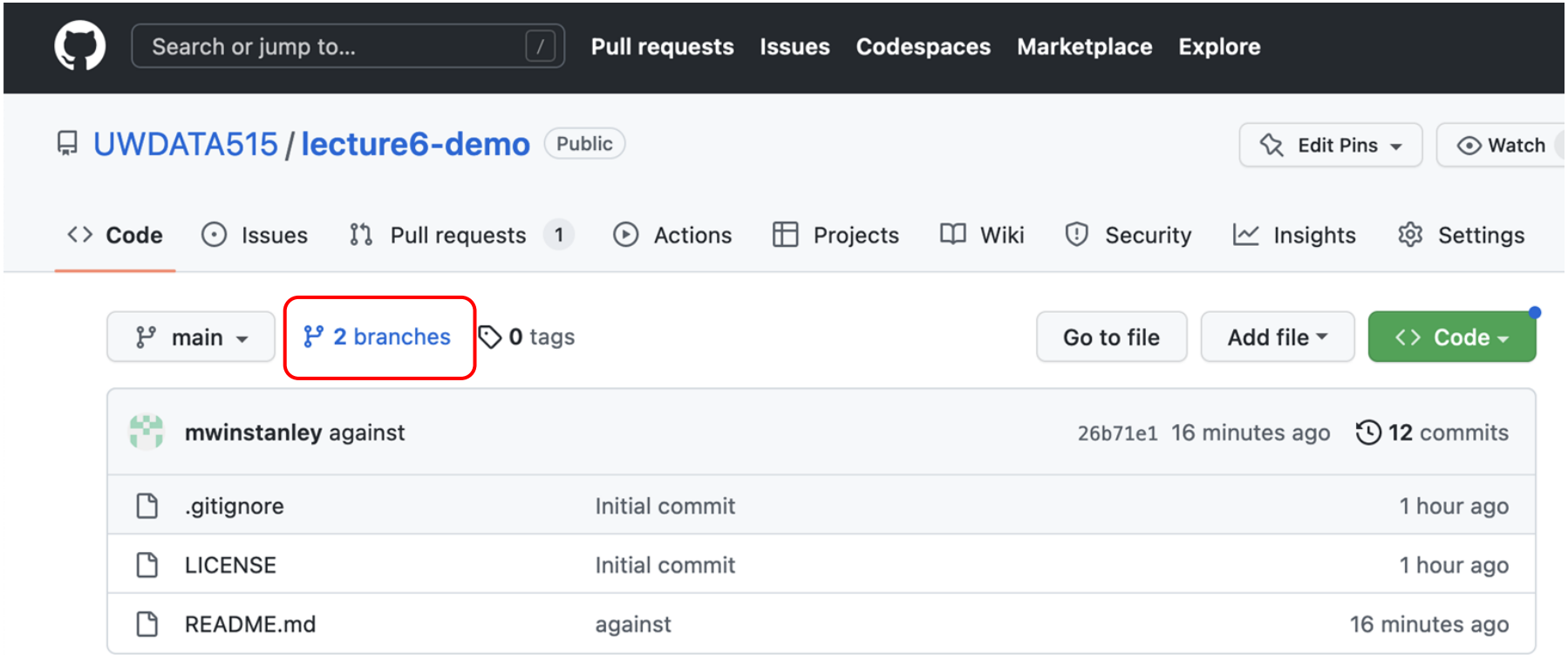
```
git config --global --add --bool push.autoSetupRemote true
```

This tells git to automatically sync the proper branch with GitHub when you push

Follow the link:

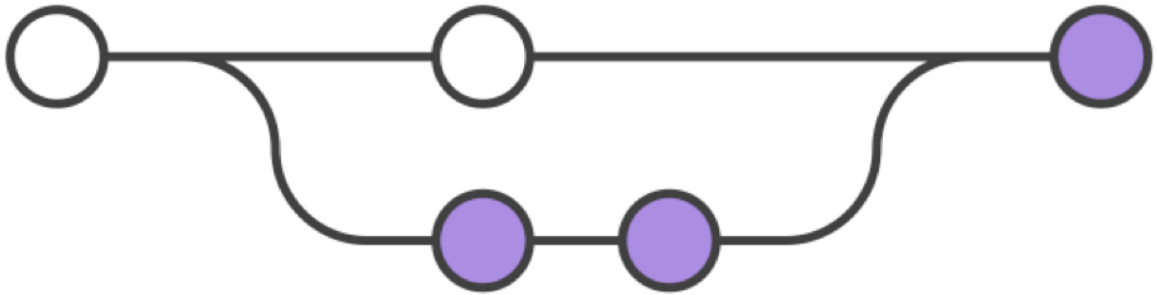
```
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
remote:
remote: Create a pull request for 'feature1' on GitHub by visiting:
remote:      https://github.com/UWDATA515/lecture6-demo/pull/new/feature1
remote:
```

Branches on GitHub



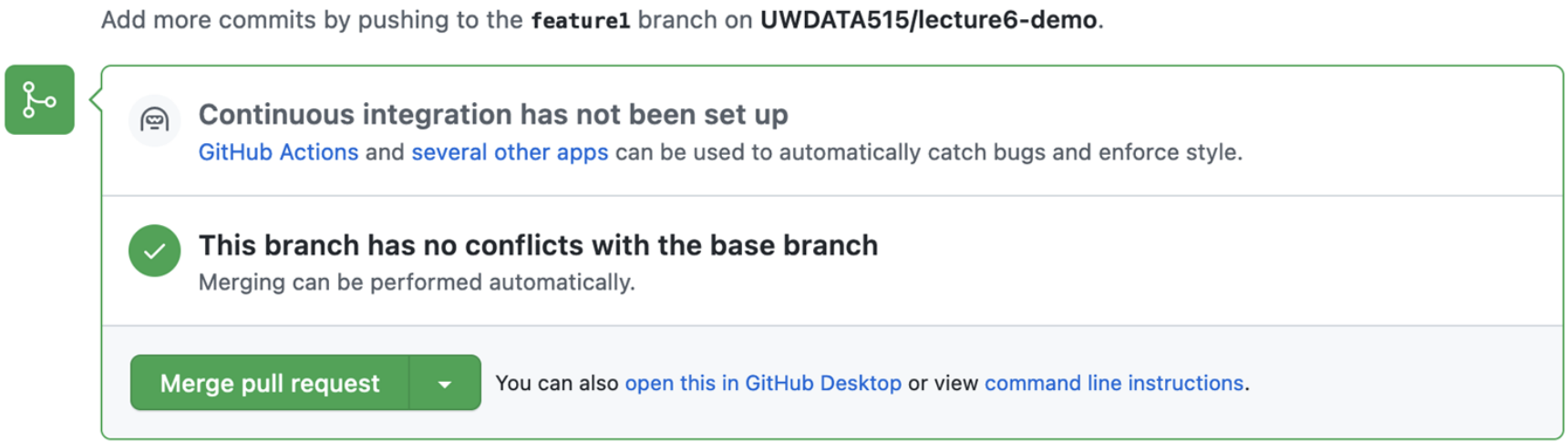
Merges

Usually: via the GitHub interface in a Pull Request (more in next slide!)



Or: `git merge branch-name` *(familiar?)*

Merging through the GitHub UI for a Pull Request



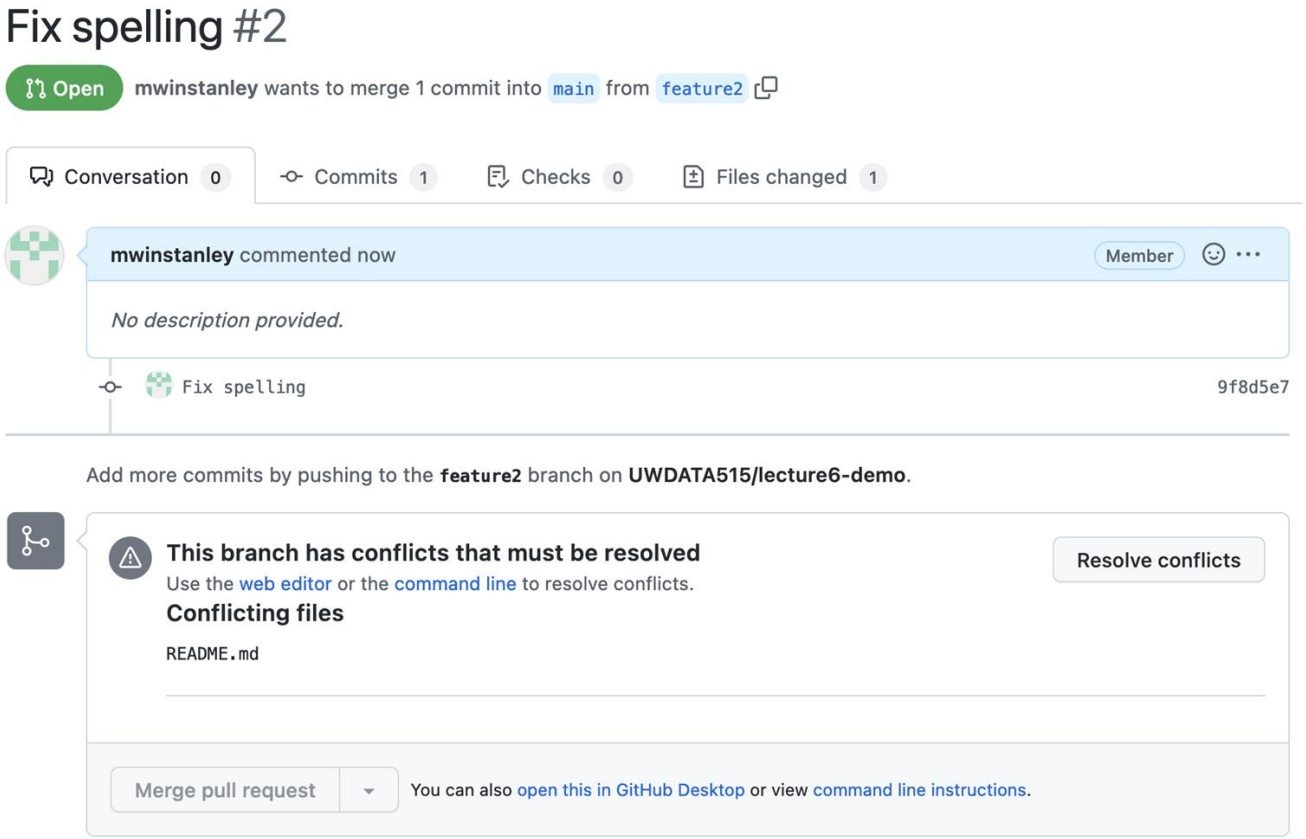
Don’t forget to delete the branch when you’re done!

(or configure auto-deletion in the repo Settings)

Merge conflicts

Uh oh!

Someone else merged a conflicting change so I can't merge my branch



Merge conflicts: merging main into your branch

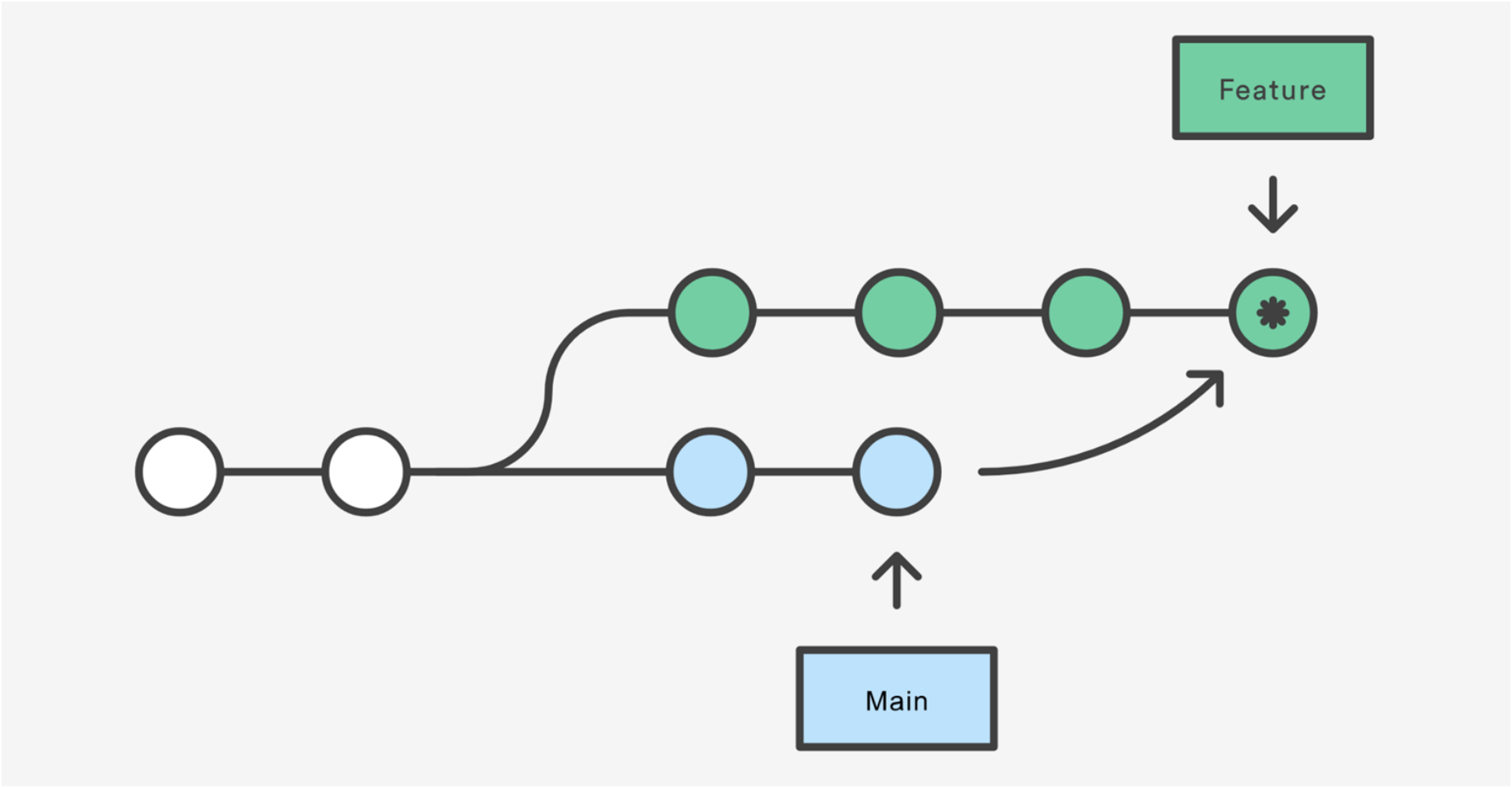
- Via the GitHub interface (ONLY if simple eg Markdown files)

Or

```
git checkout main
git pull              # pull in the conflict
git checkout your-feature-branch
git merge main
```

(fix merge conflicts by editing the conflicting files, git add as needed)

```
git commit
git push              # update the remote version of your branch
```


Feature
Main

Merging main into your branch

Pros

- Non-destructive - existing branches are not changed in any way

Cons

- Extra “merge commits” in the history

Merge conflicts: rebasing

If you just HATE those “merge commits”

```
git checkout main
git pull                    # pull in the conflict
git checkout your-feature-branch
git rebase main
```

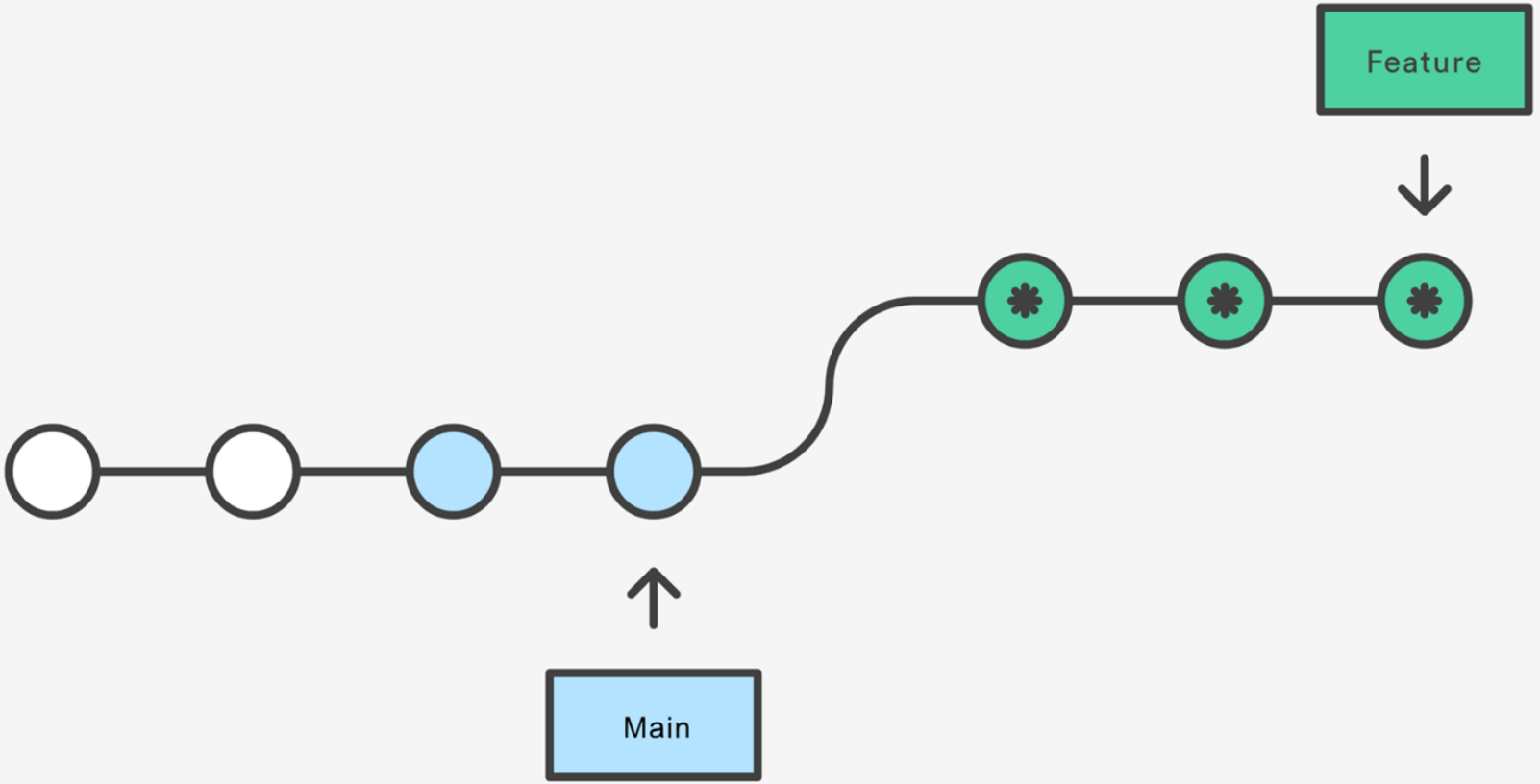
(fix merge conflicts by editing the conflicting files, git add as needed)

```
git rebase --continue
git push --force
```

This “rewrites history” - hence why you need to `--force`

NEVER rebase main (or any branch that multiple people work on at once)!!!!

More about merging vs rebasing


Feature
Main

Rebasing

Pros

- Perfectly linear history
- No extraneous “merge commits” - cleaner history

Cons

- Rewriting history
- You can’t tell when things were rebased exactly

NEVER USE REBASING ON A PUBLIC BRANCH (ie main)

Only your own private branches

Rebasing

- You don’t have to wait until you have an approval to rebase or merge main into your branch!
- It’s a good idea if you know someone else submitted a major change that might affect yours

Standard Collaborative Flow

- Create a new branch (`git checkout -b new-branch-name`)
- Commit some changes
- Push your branch to GitHub
- Create a pull request in the UI
- Pull and merge main into your branch again, resolving conflicts, if necessary (`git merge main` or `git rebase main`)
- Merge the pull request using github.com and the big green button

Exercise

Everyone:

Follow the standard collaborative flow to make a change to README.md

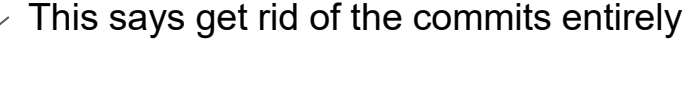
Review each other's pull requests

Everyone should merge

Common mistake

Oops! I forgot to switch to a new branch and now I my commits are on `main`!

- Create a new branch where you are. This will save your commits and give then the new branch name
- Checkout the main branch
- Reset main back to before your commits. Count how many commits back you want to go, then reset. For example, if you've accidentally added 2 commits:




```
git reset --hard HEAD~2
```

- Now you can checkout your new branch and continue with your changes

Real-World Branch Notes

- Conflicts are natural! There is nothing wrong
- Name your branch something descriptive
- Prefix your branch name with your user id

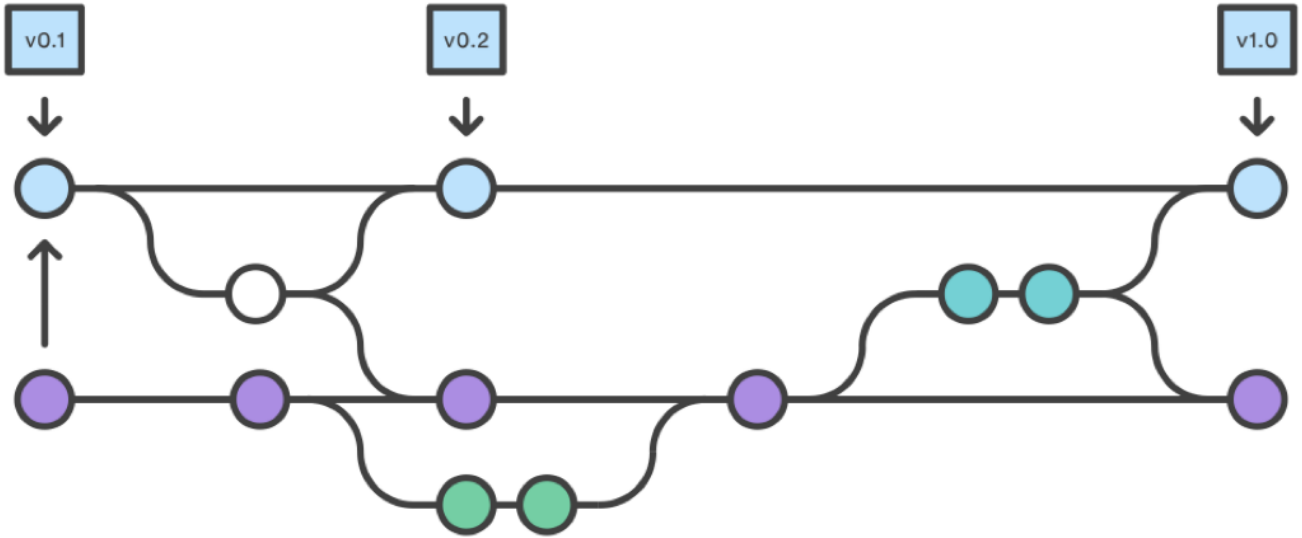
  `naomila/readmeUpdate`
- Delete your branch when you're done
  - Configure auto-deletion in the repository
  - Settings > Automatically delete head branches

Forking

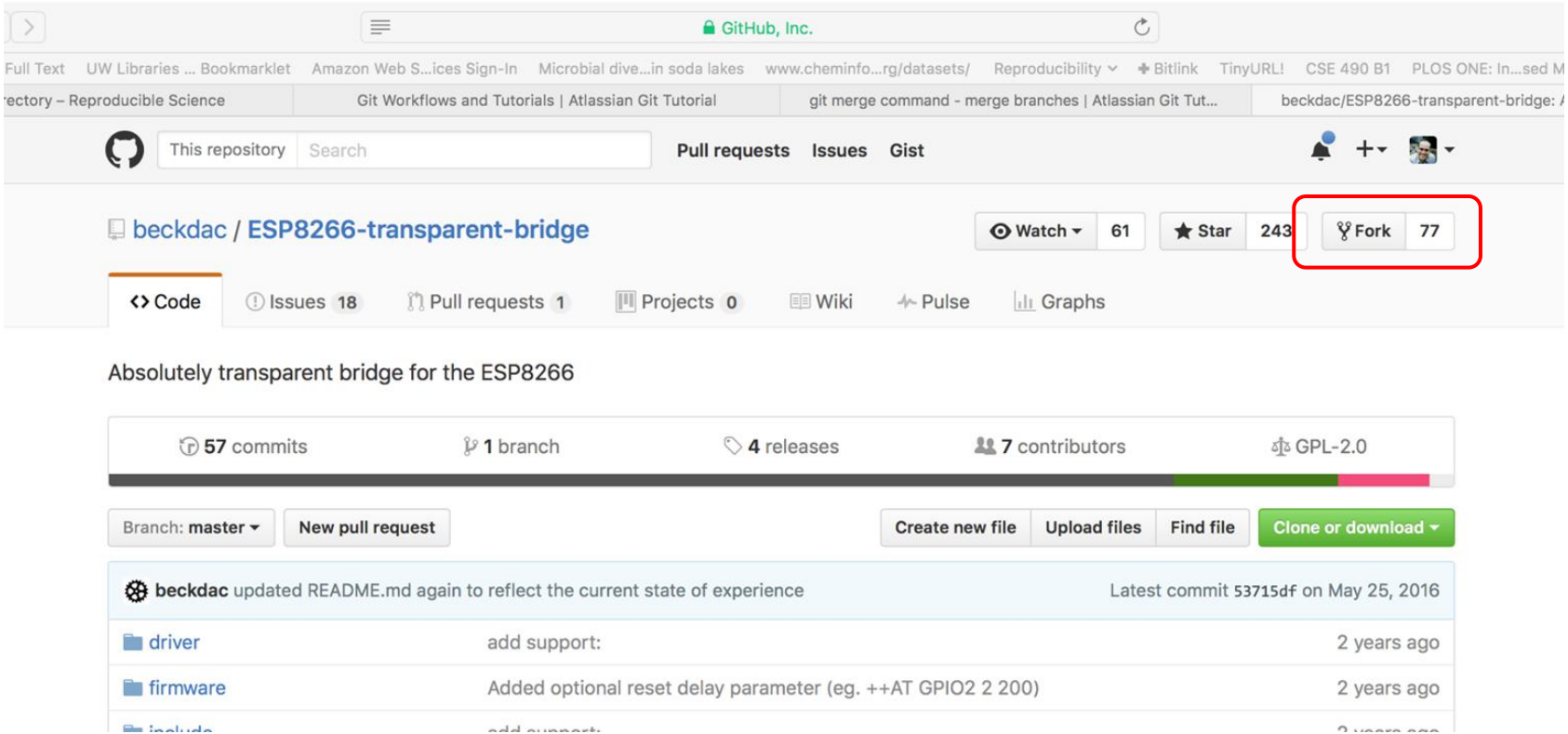
Another solution!

Forking

A totally new copy of the repository, which you can change and ask the original owners to “pull” back into their own version.



Forking


GitHub, Inc.
Full Text
UW Libraries ... Bookmarklet
Amazon Web S...ices Sign-In
Microbial dive...in soda lakes
www.cheminfo...rg/datasets/
Reproducibility
Bitlink
TinyURL!
CSE 490 B1
PLOS ONE: In...sed M
ectory – Reproducible Science
Git Workflows and Tutorials | Atlassian Git Tutorial
git merge command - merge branches | Atlassian Git Tut...
beckdac/ESP8266-transparent-bridge:
This repository
Search
Pull requests
Issues
Gist
beckdac / ESP8266-transparent-bridge
Watch 61
Star 243
Fork 77
Code
Issues 18
Pull requests 1
Projects 0
Wiki
Pulse
Graphs
Absolutely transparent bridge for the ESP8266
57 commits
1 branch
4 releases
7 contributors
GPL-2.0
Branch: master
New pull request
Create new file
Upload files
Find file
Clone or download
beckdac updated README.md again to reflect the current state of experience
Latest commit 53715df on May 25, 2016
driver
add support:
2 years ago
firmware
Added optional reset delay parameter (eg. ++AT GPIO2 2 200)
2 years ago

Forking

Probably not in this class (use branches)!

Making changes to open-source libraries