

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?

Shane



Ilya



Figure: Multiple versions can be merged

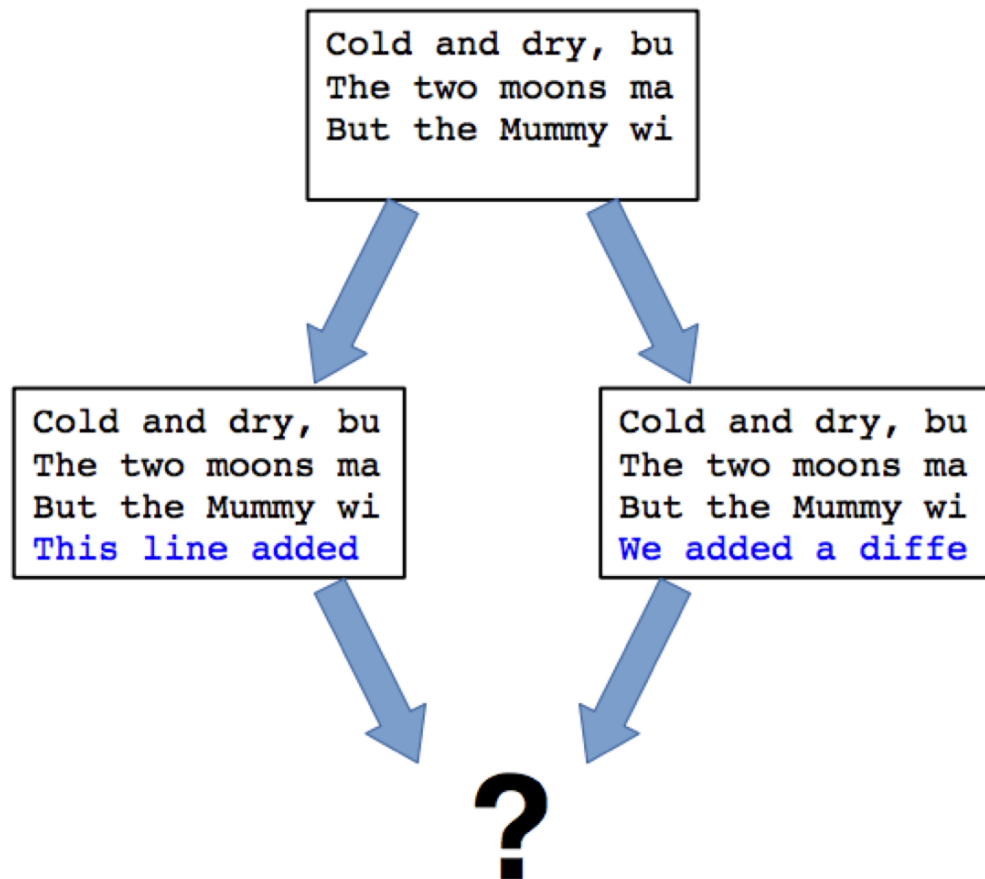


Figure: The conflicting changes

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

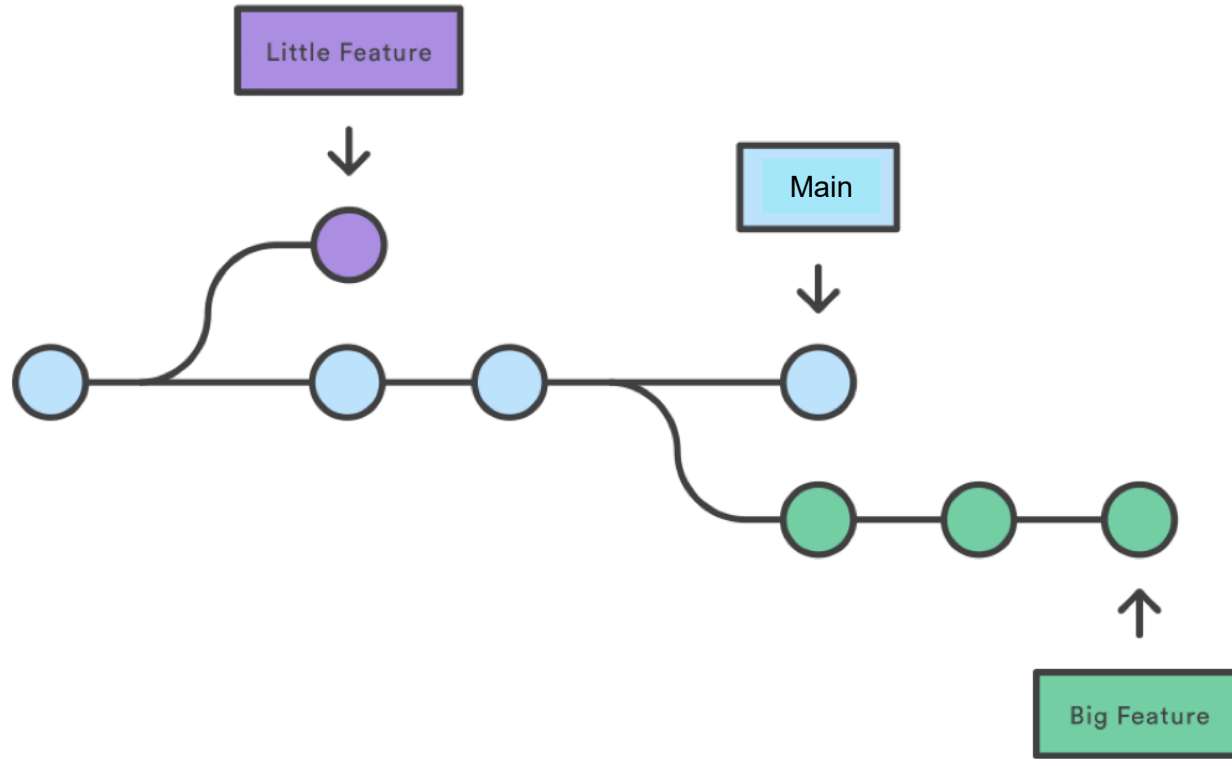
```
git add the file
```

```
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)

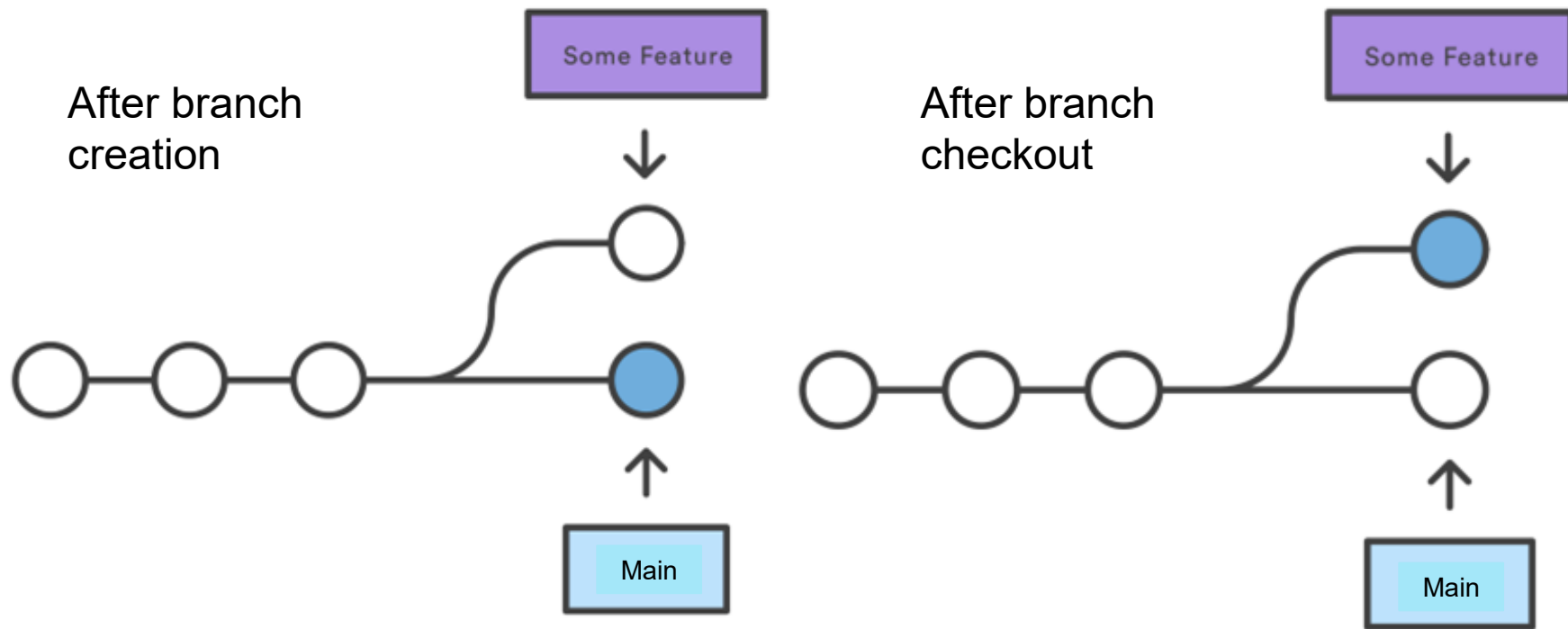
```
git checkout name-of-branch
```

```
git checkout main
```

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

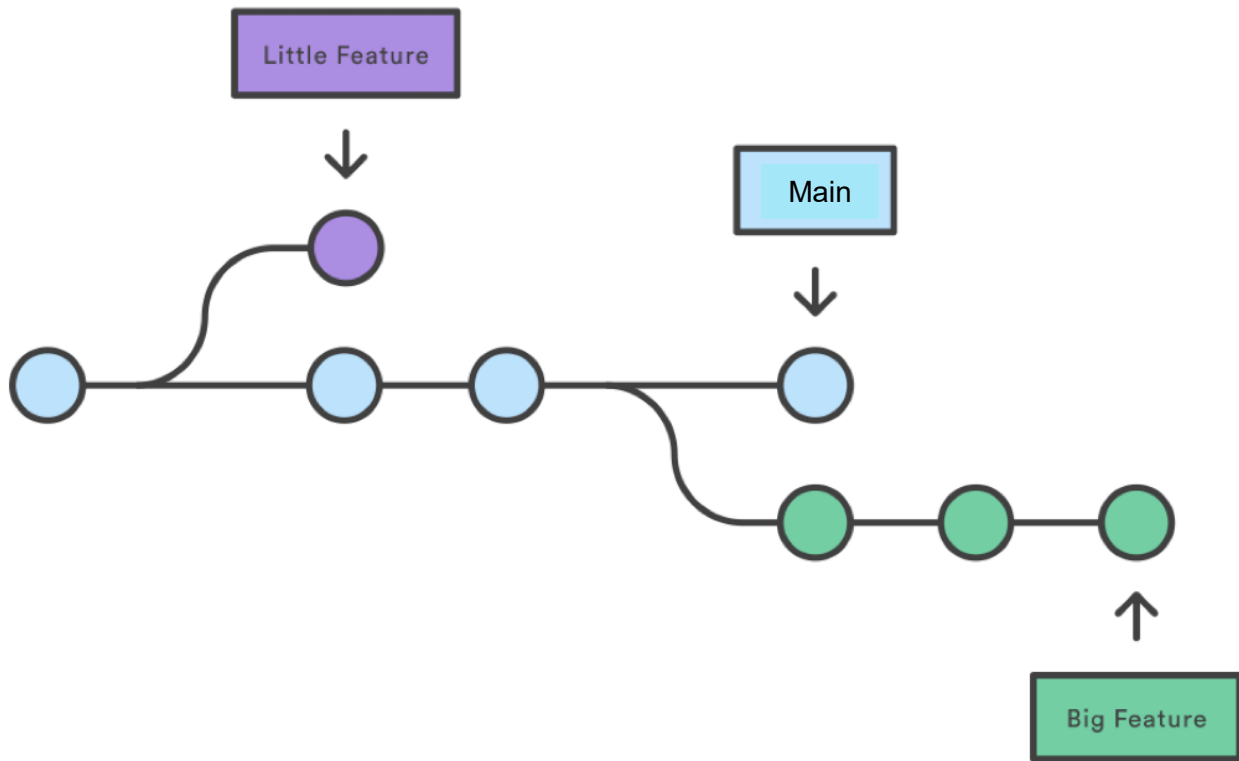
```
git checkout -b name-of-new-branch
```

Branches: visual





Merges


But how do we
get our features
back into main?



Pull requests

 Search or jump to... 

[Pull requests](#) [Issues](#) [Codespaces](#) [Marketplace](#) [Explore](#)


 [graphql / graphql-js](#) Public Edit Pins Watch 402


[Code](#) [Issues 143](#) [Pull requests 99](#) [Discussions](#) [Actions](#) [Projects 1](#) [Wiki](#) [Security](#)

TS: Fix incorrect enum typing in findBreakingChanges #2563

[Merged](#) IvanGoncharov merged 1 commit into [graphql:master](#) from [mwinstanley:mwinstanley/fixBreakingChangesTyping](#) on M



[Conversation 1](#) [Commits 1](#) [Checks 0](#) [Files changed 1](#)





mwinstanley commented on May 18, 2020 Contributor 

PR #2084 ([link to relevant part](#)) updated the enums used in `findBreakingChanges.js`, but did not update the corresponding types in `findBreakingChanges.d.ts`. Found this inconsistency when playing around with `findBreakingChanges` in Typescript.

This PR updates the typing to correspond with the implementation.

  TS: Fix incorrect enum typing in findBreakingChanges cfeb36a

  IvanGoncharov added the [PR: bug fix](#) label on May 18, 2020

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





[Pull requests](#) [Issues](#) [Codespaces](#) [Marketplace](#) [Explore](#)

 [UWDATA515 / lecture6-demo](#) Public Edit Pins Watch

[Code](#) [Issues](#) [Pull requests](#) 1 [Actions](#) [Projects](#) [Wiki](#) [Security](#) [Insights](#) [Settings](#)

 main


 2 branches




 0 tags

[Go to file](#)

[Add file](#)

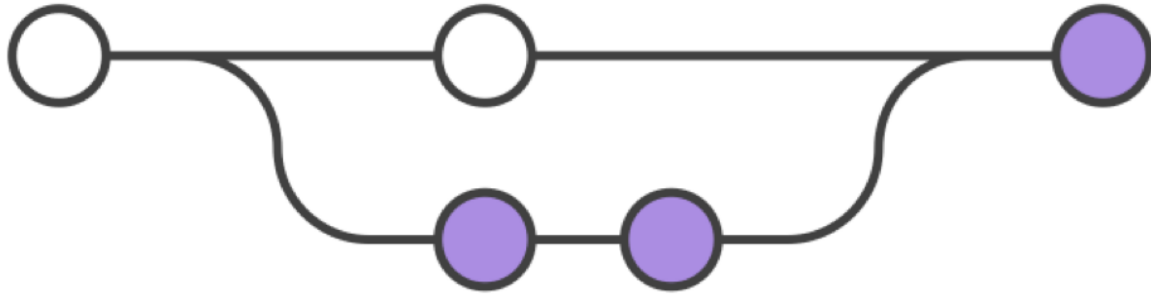
[Code](#)

 **mwinstanley** against 26b71e1 16 minutes ago 12 commits

 .gitignore	Initial commit	1 hour ago
 LICENSE	Initial commit	1 hour ago
 README.md	against	16 minutes ago

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

Add more commits by pushing to the **feature1** branch on **UWDATA515/lecture6-demo**.






Continuous integration has not been set up
[GitHub Actions](#) and [several other apps](#) can be used to automatically catch bugs and enforce style.



This branch has no conflicts with the base branch
Merging can be performed automatically.

Merge pull request 

You can also [open this in GitHub Desktop](#) or view [command line instructions](#).

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

Fix spelling #2

 Open mwinstanley wants to merge 1 commit into `main` from `feature2` 

 Conversation 0

 Commits 1

 Checks 0

 Files changed 1



mwinstanley commented now

Member



No description provided.



Fix spelling

9f8d5e7

Add more commits by pushing to the `feature2` branch on `UWDATA515/lecture6-demo`.



This branch has conflicts that must be resolved

Use the [web editor](#) or the [command line](#) to resolve conflicts.

Conflicting files

README.md

Resolve conflicts

Merge pull request



You can also [open this in GitHub Desktop](#) or view [command line instructions](#).

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
```

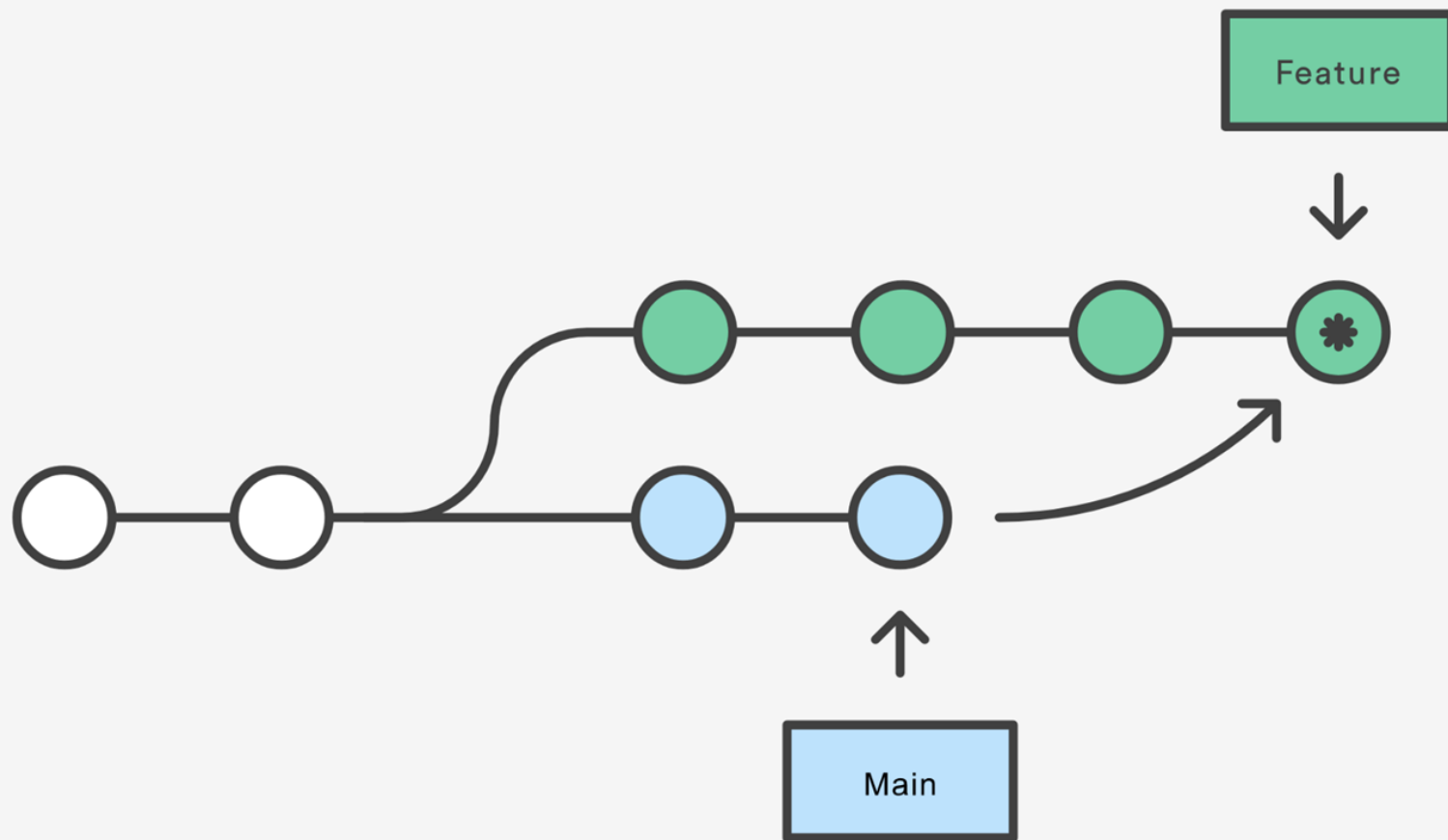
```
git checkout your-feature-branch
```

```
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
```



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)

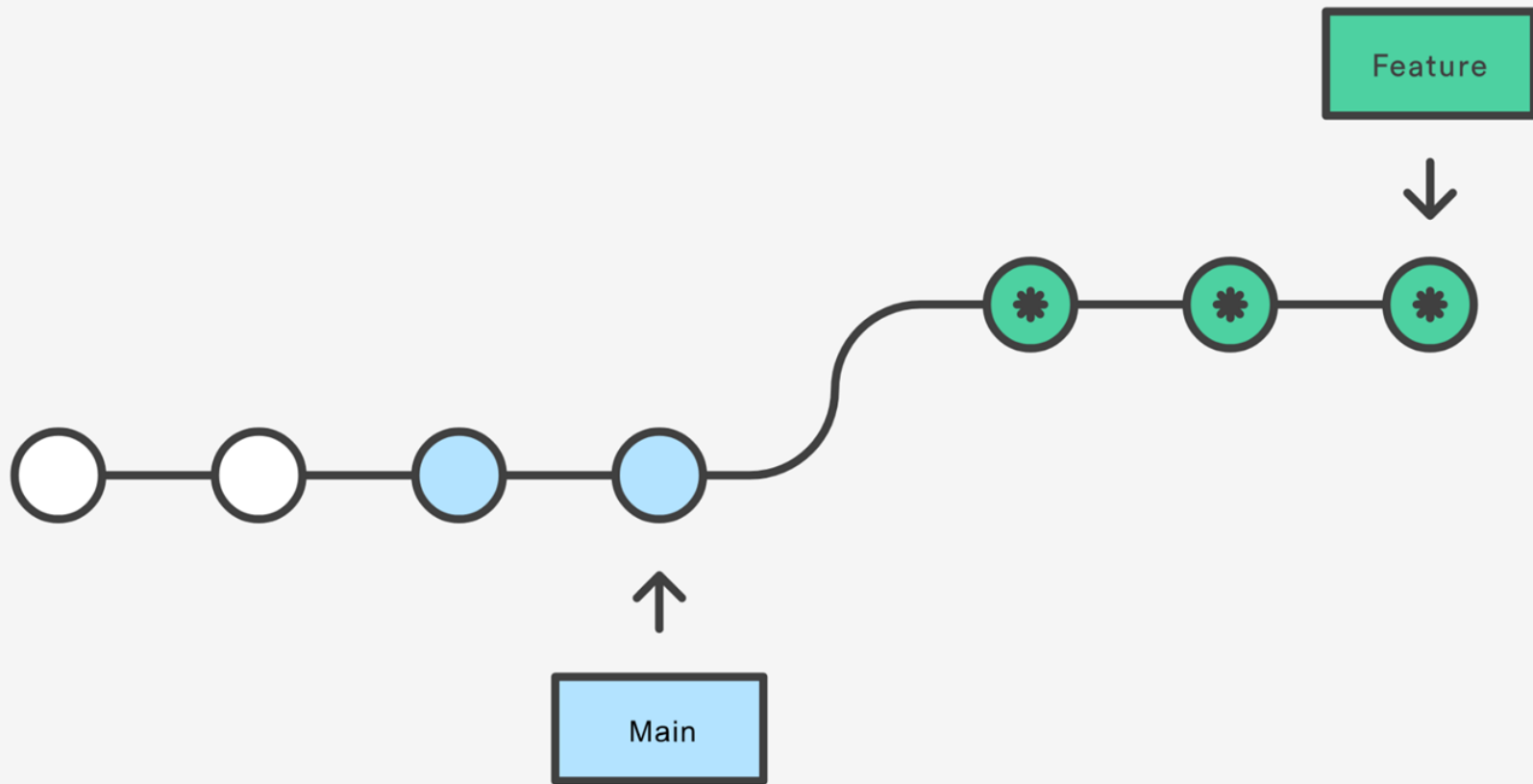
```
git rebase --continue
```

```
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](#)



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
```

↙ This says get rid of the commits entirely

- 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

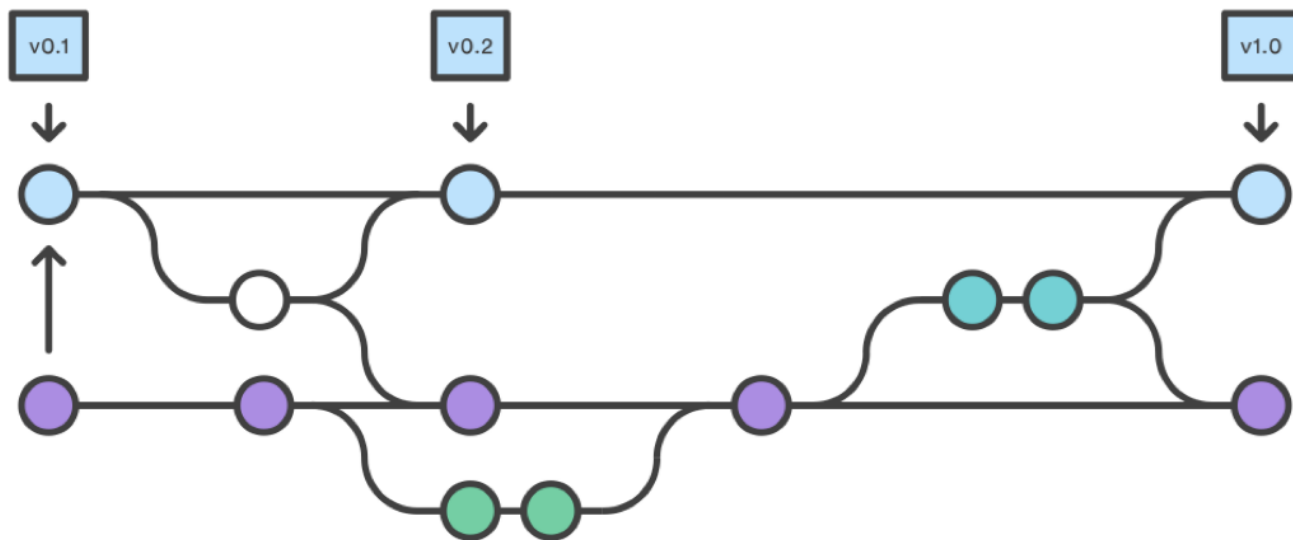
Forking

Another
solution!

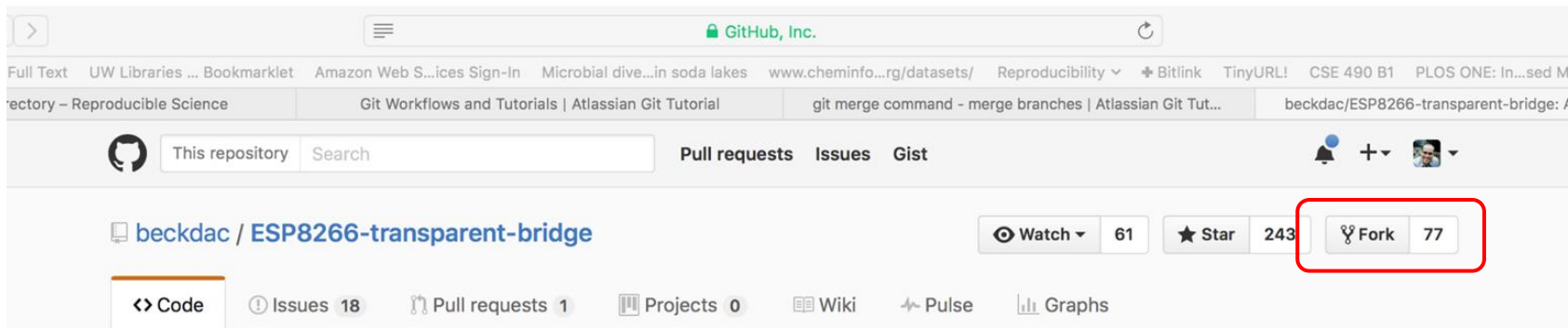


Forking

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

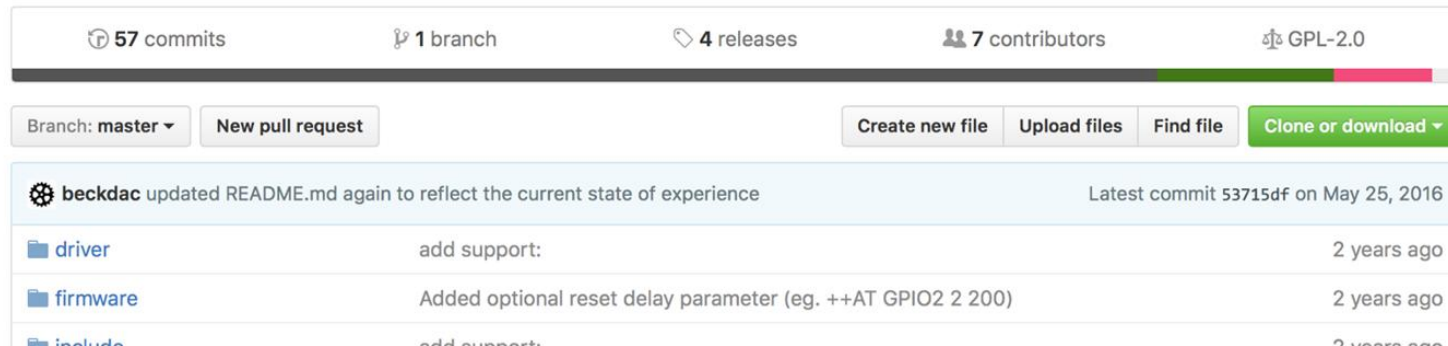


Forking



The screenshot shows the GitHub interface for the repository `becdac / ESP8266-transparent-bridge`. At the top, there's a navigation bar with the GitHub logo, a search bar, and links for Pull requests, Issues, and Gist. Below this, the repository name is displayed. To the right of the repository name, there are buttons for Watch (61), Star (243), and Fork (77). The Fork button is highlighted with a red box. Below the repository name, there are tabs for Code, Issues (18), Pull requests (1), Projects (0), Wiki, Pulse, and Graphs.

Absolutely transparent bridge for the ESP8266



The screenshot shows the repository details section. It includes a progress bar and statistics: 57 commits, 1 branch, 4 releases, 7 contributors, and GPL-2.0 license. Below this, there are buttons for Branch: master, New pull request, Create new file, Upload files, Find file, and Clone or download. A commit message is displayed: "becdac updated README.md again to reflect the current state of experience" with the latest commit hash 53715df on May 25, 2016. Below the commit message, there is a list of files: driver, firmware, and include, each with a description of the changes and the time since the last commit.

File	Description	Time
driver	add support:	2 years ago
firmware	Added optional reset delay parameter (eg. ++AT GPIO2 2 200)	2 years ago
include	add support:	2 years ago

Forking

Probably not in this class (use branches)!

Making changes to open-source libraries