

Software Design for Data Science

Git & Team Collaboration

*Melissa Winstanley
University of Washington
February 7, 2023*



Project Step 1: Due next Tuesday, 2/14

- Create a GitHub repository
- Validate your idea with your team
- Add a README.md file with a project intro
 - Project type
 - Questions of interest
 - Goal for the project output (what are you going to produce?)
 - Data sources you will use
- Create your team in Canvas > People > Project Teams
- One person to submit the repository link via Canvas

Idea Validation

- Agree as a team on what the project is
 - Clarity about the project type
 - Consensus on the problem being solved
- Validate that the project is feasible and large enough
 - Is there an unmet need (i.e. no code already exists)?
 - Do you have data that can solve the problem?
 - Will this project take about a month of effort for 3-4 people to complete?

Intermediate Git

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?

Mary



Alice



Figure: Multiple versions can be merged

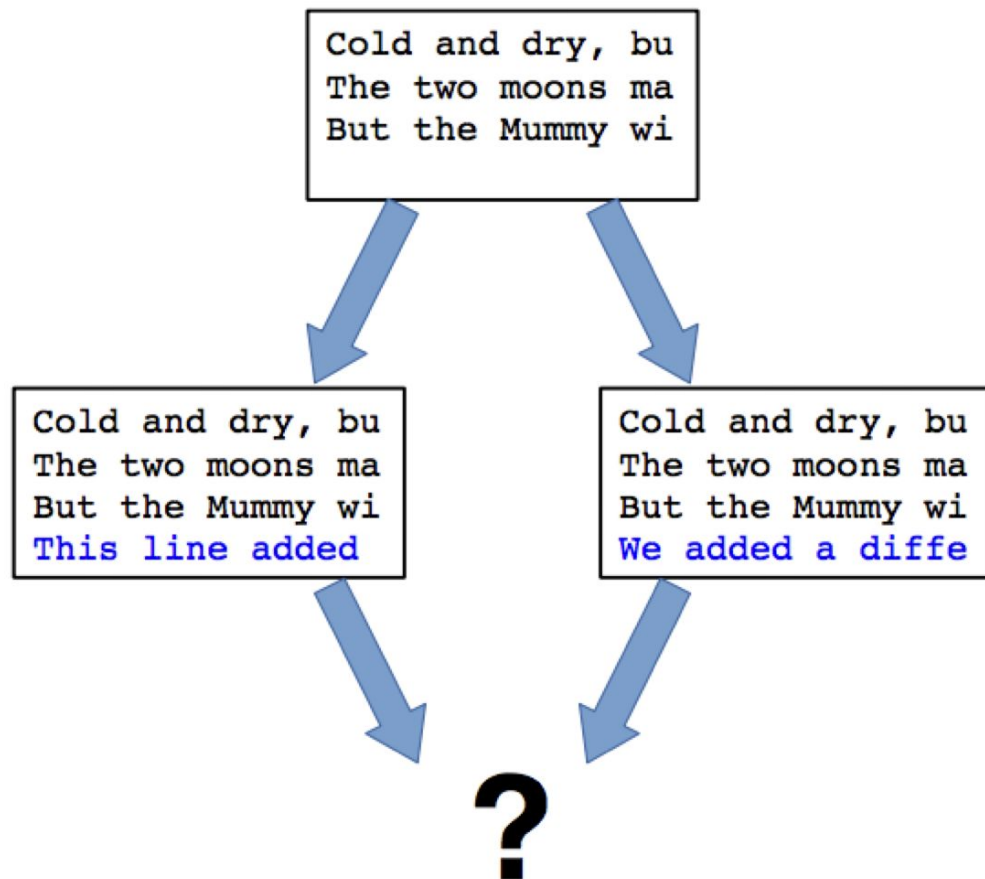


Figure: The conflicting changes

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
```

Open the files with the conflict in a text editor

Find the conflict, which always has lots of <<<<< and >>>>>

Rewrite the block between the brackets to what you want it to be

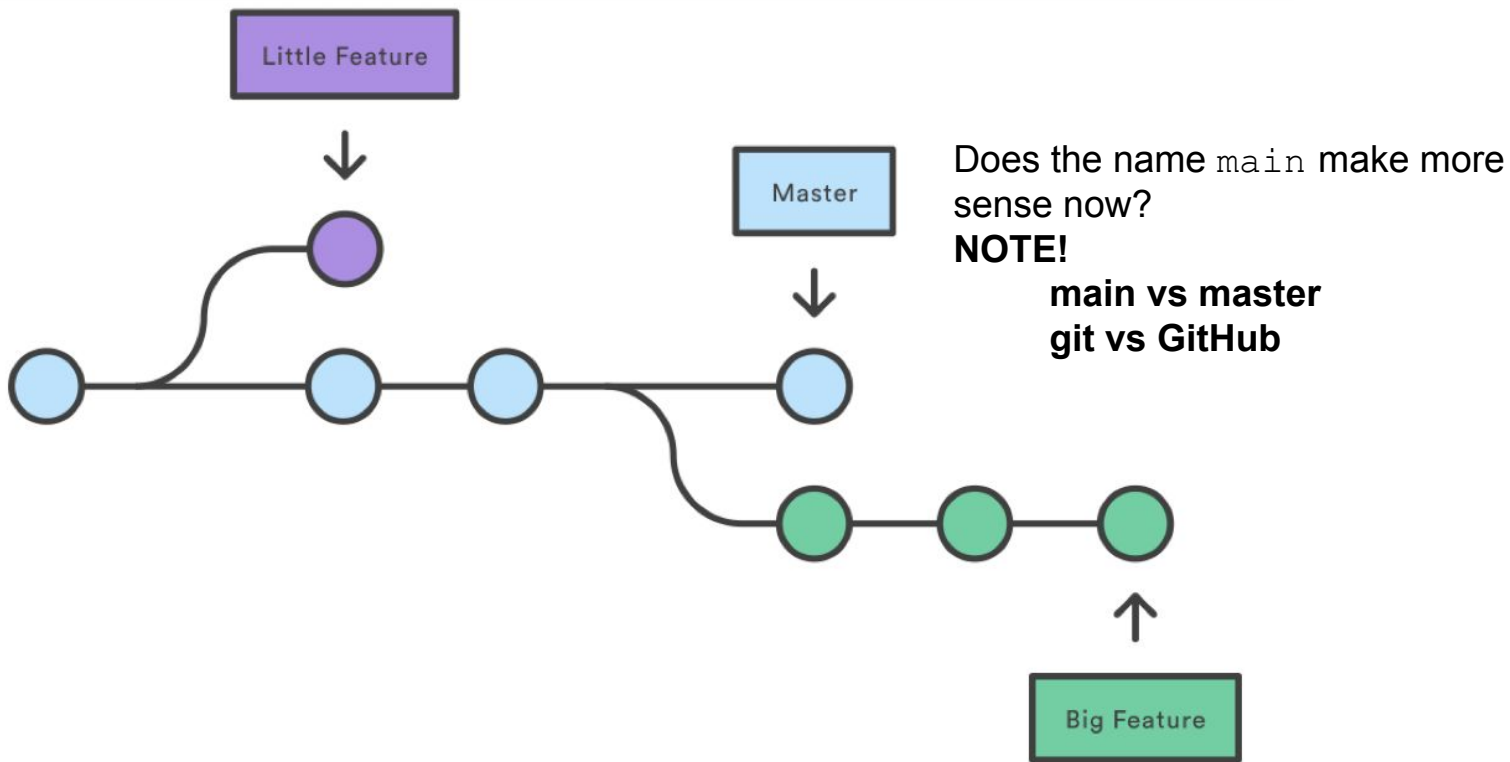
Remove the brackets and equals signs too

```
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)

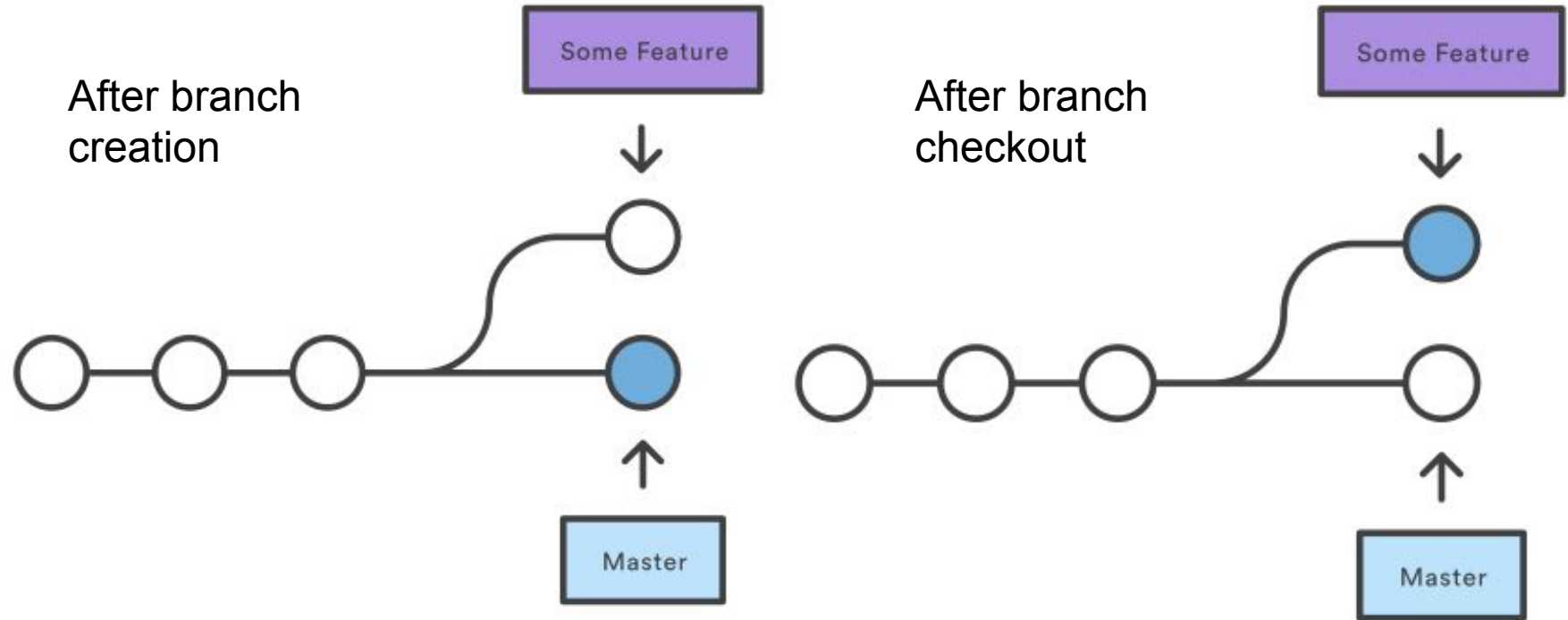
```
git checkout name-of-branch
```

```
git checkout main
```

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

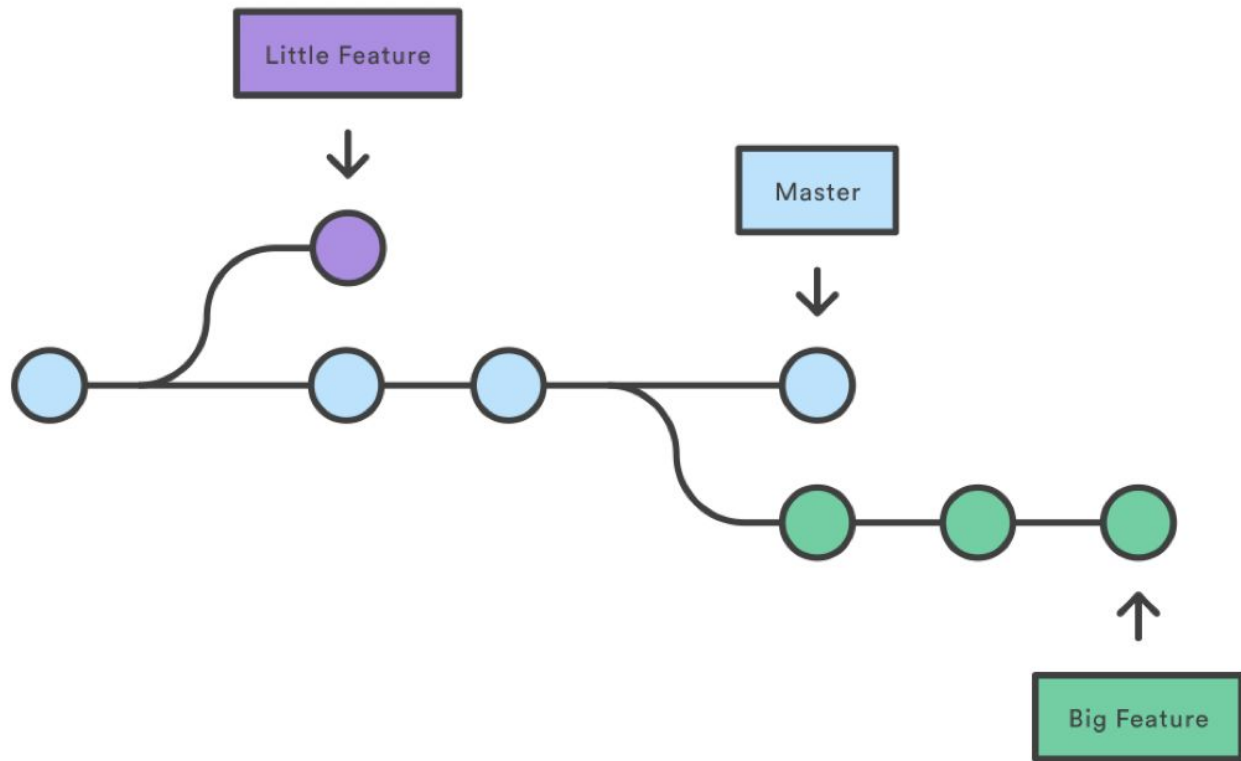
```
git checkout -b name-of-new-branch
```

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


 Search or jump to... 




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

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

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

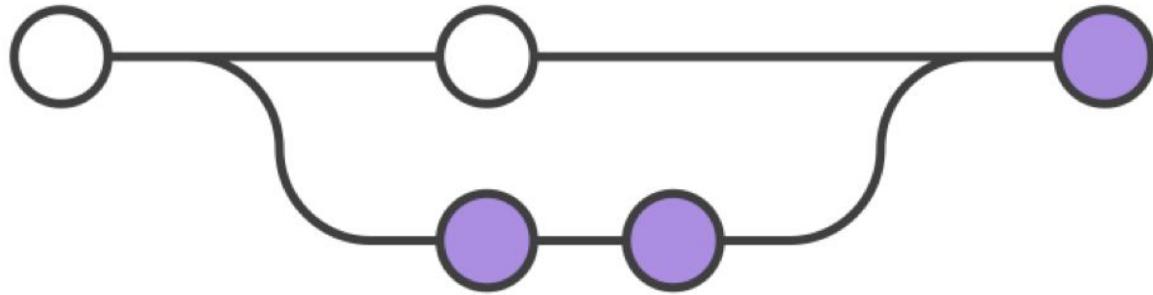
[main](#) [2 branches](#) [0 tags](#) [Go to file](#) [Add file](#) [Code](#)

 **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

Resolve conflicts

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

Conflicting files

README.md

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
```

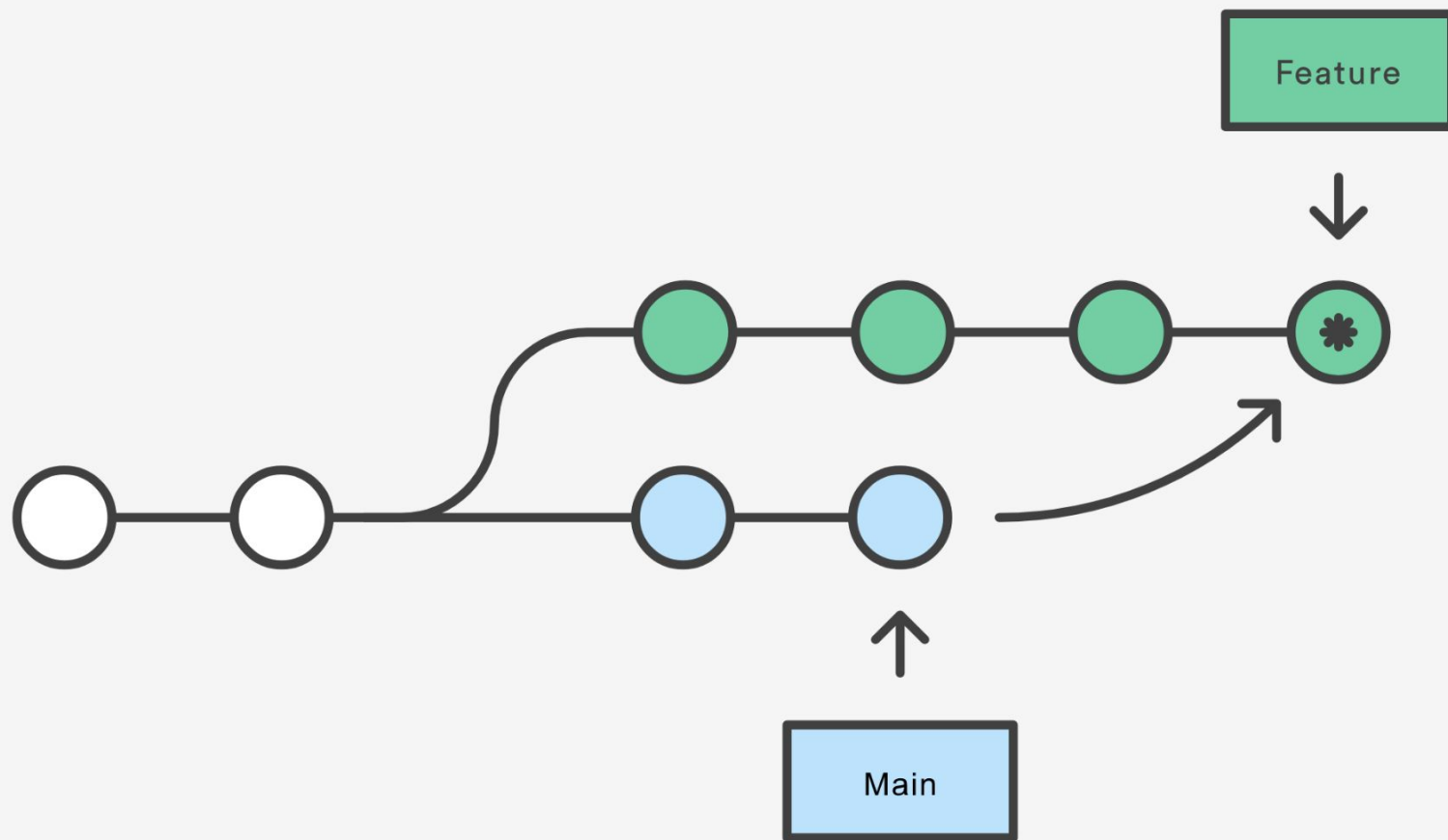
```
git checkout your-feature-branch
```

```
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're like me and HATE those “merge commits”

```
git rebase main
```

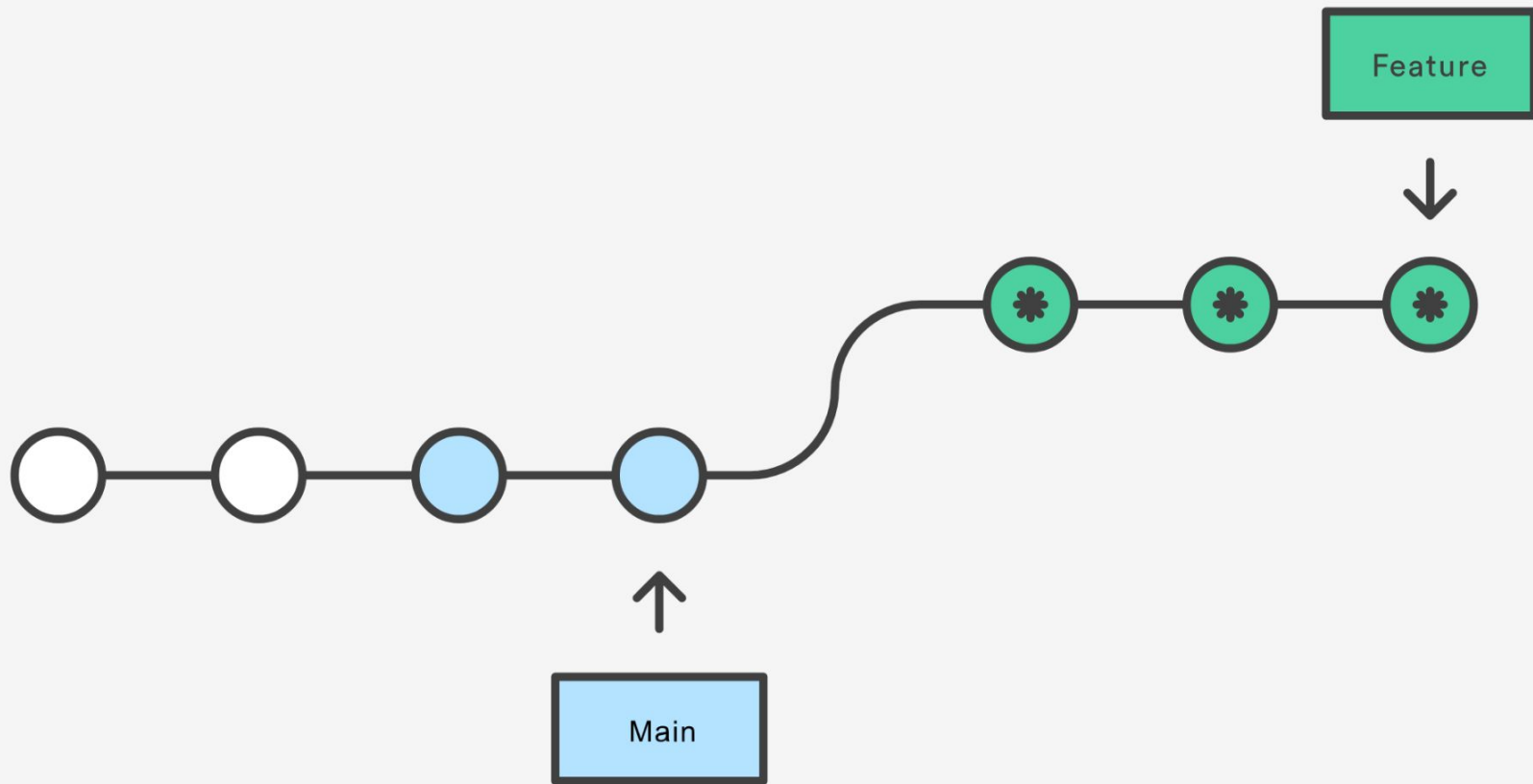
(fix merge conflicts by editing the conflicting files, git add as needed)

```
git rebase --continue
```

```
git push --force
```

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

[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
- Commit some changes
- Push your branch to GitHub
- Create a pull request
- Pull and merge main into your branch again, resolving conflicts, if necessary
- 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

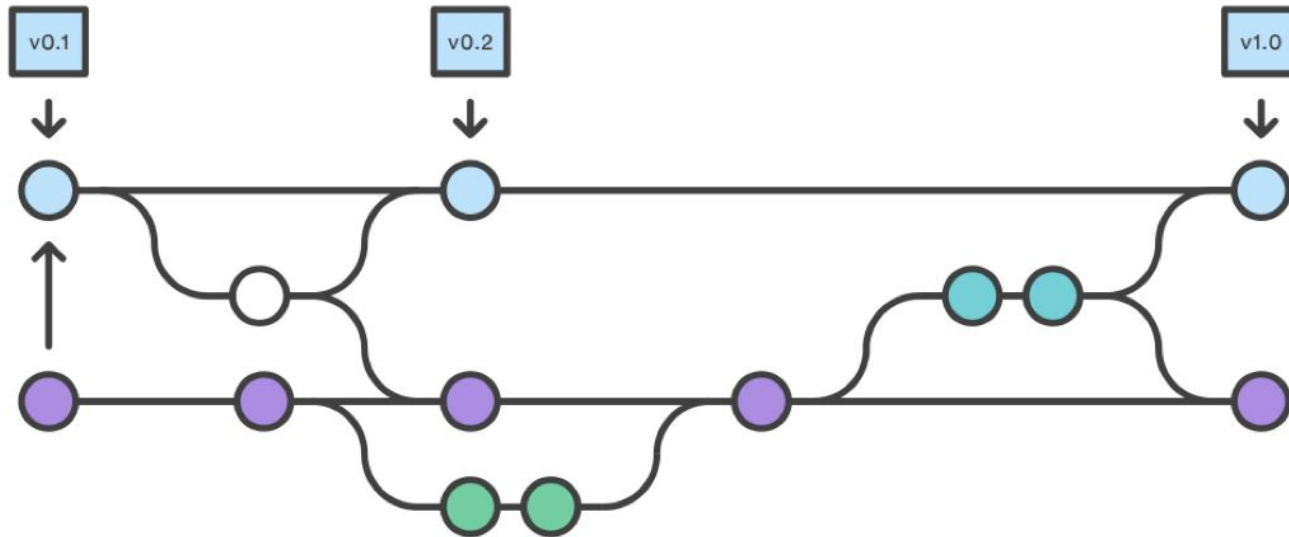
Forking

Another
solution!

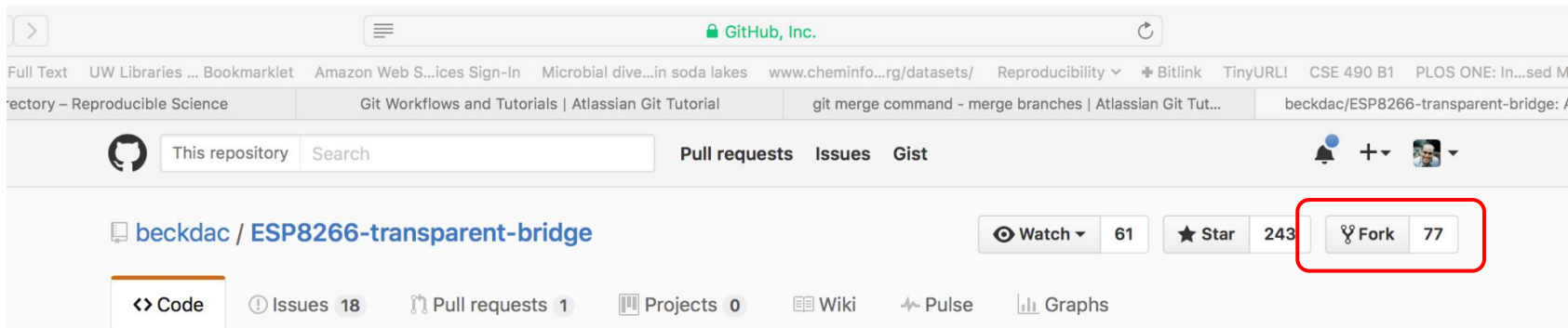


Forking

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

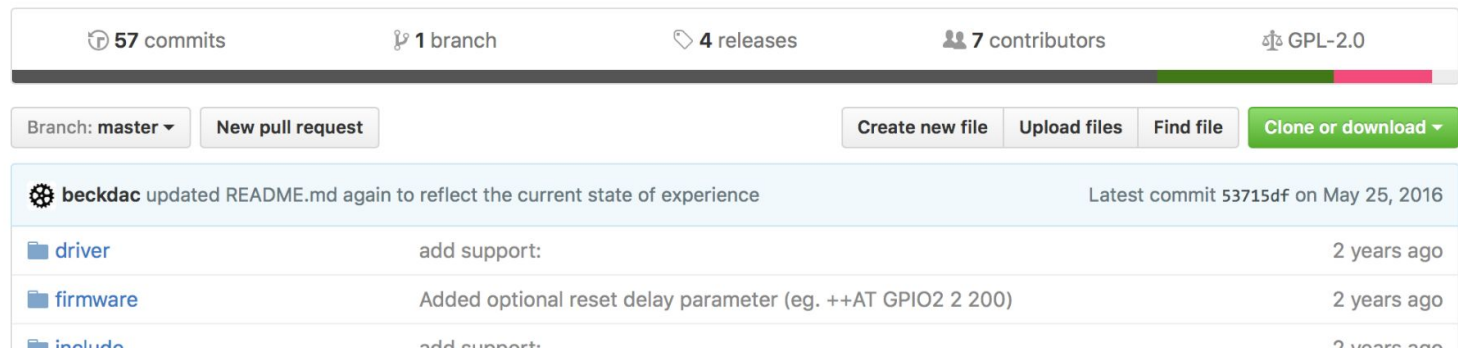


Forking



The screenshot shows the GitHub repository page for `beckdac / ESP8266-transparent-bridge`. The repository has 61 watches, 243 stars, and 77 forks. 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 statistics and commit history for `ESP8266-transparent-bridge`. The statistics bar indicates 57 commits, 1 branch, 4 releases, 7 contributors, and GPL-2.0 license. Below the statistics, there are buttons for 'Branch: master', 'New pull request', 'Create new file', 'Upload files', 'Find file', and 'Clone or download'. The commit history shows the latest commit by `beckdac` updating the README.md, dated May 25, 2016. Below the commit history, there is a table listing files and their commit messages.

File	Commit Message	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

Team Collaboration



Standups

- What is a standup?
 - 1-2 minutes per person
- Why standups?
 - Communicate status and actions within and between teams
- What do I say in a standup?
 - Progress you've made since the last standup
 - How it compares with the plan
 - If behind plan, how to compensate to make plan end date
 - Deliverables for next period
 - Challenges to making next deliverables
 - Technology uncertainties and blockers
 - Team issues


Standups

In class!

GitHub Issues


 Search or jump to... 




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

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

[Code](#) [Issues](#) [Pull requests](#) [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

GitHub Issues

- Good way to keep track of
 - Milestones
 - Tasks
 - Bugs
- If you link to an issue in a PR, it will auto-link the two
- Use it if you like!

GitHub Pull Requests

- Pull Request = “PR”
 - Yeah, it’s a confusing name: “hey repo owners, please PULL my change into main”
- What’s in the description of a PR?
 - Background
 - Describe what change you are making
 - Describe how that change affects the application
 - Test plan
 - How did you test the code?
 - Include specific commands that you ran, and the output
- ALWAYS!
- Only push directly to main in a true emergency (not in this class!)

Good option: draft pull requests

If you're not fully ready to submit it for code review



Another update

Write Preview

H B I

Leave a comment

Attach files by dragging & dropping, selecting or pasting them.

Create pull request



Remember, contributions to this repository should follow our [GitHub Community Guidelines](#)



Create pull request

Open a pull request that is ready for review

Create draft pull request

Cannot be merged until marked ready for review

Code Reviews

- What is code review?
 - Someone looks at, comments on, and approves a PR
- Why code review?
 - To find bugs
 - To improve code quality
- What do you comment on in a code review?
 - Bugs or missing pieces
 - Style
 - Choice of variable and function names
 - Code readability
 - If the PR doesn't have a test plan
 - How to improve reuse and efficiency
 - How to use existing Python packages

Branch Protection

- Protect `main` against accidents!
- Require pull requests and code reviews
- Common in the real world

```
remote: error: GH006: Protected branch update failed for refs/heads/main.  
remote: error: At least 1 approving review is required by reviewers with write access.  
To https://github.com/UWDATA515/lecture6-demo.git  
! [remote rejected] main -> main (protected branch hook declined)  
error: failed to push some refs to 'https://github.com/UWDATA515/lecture6-demo.git'
```


Branch Protection

GitHub Repository Page > Settings > Branches > Add branch protection rule

Branch name pattern *

main

Protect matching branches

☒ **Require a pull request before merging**

When enabled, all commits must be made to a non-protected branch and submitted via a pull request before they can be merged into a branch that matches this rule.

☒ **Require approvals**

When enabled, pull requests targeting a matching branch require a number of approvals and no changes requested before they can be merged.

Required number

☒ **Do not allow bypassing the above settings**

The above settings will apply to administrators and custom roles with the "bypass branch protections" permission.

Do this now for
your exercise
repository!
Do this on your
project repository!

Communication strategies

- Use a chat mechanism for informal, quick communication
 - Email doesn't really work
 - Slack
 - Microsoft Teams
 - Discord
 - Google Chat
- Don't hesitate to hop on a Zoom to follow-up
- Documentation as communication
 - Code documentation
 - Pull request description

Pair programming

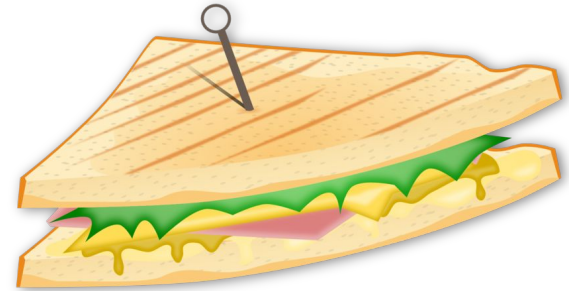
- Two people, one keyboard
- Fewer mistakes with more eyes
- Learning for both people

Consider trying it for difficult tasks!

- Swap who has hands on keyboard at least every 30 minutes
- Keep talking

Fast feedback

- Quick responses to communication
 - Code reviews
 - Emails
 - Chat
- Constructive criticism
 - Consider (gently) giving teammates feedback
 - Make observations, not generalization
 - Sandwich approach
 - Listen to the response!
 - Don't wait until the survey at the end of the quarter!



Collaboration Summary

- Standups
- GitHub issues
- Pull requests
- Code reviews
- Chat
- Pair programming
- Fast feedback