Collaborative Git and GitHub

Introductions

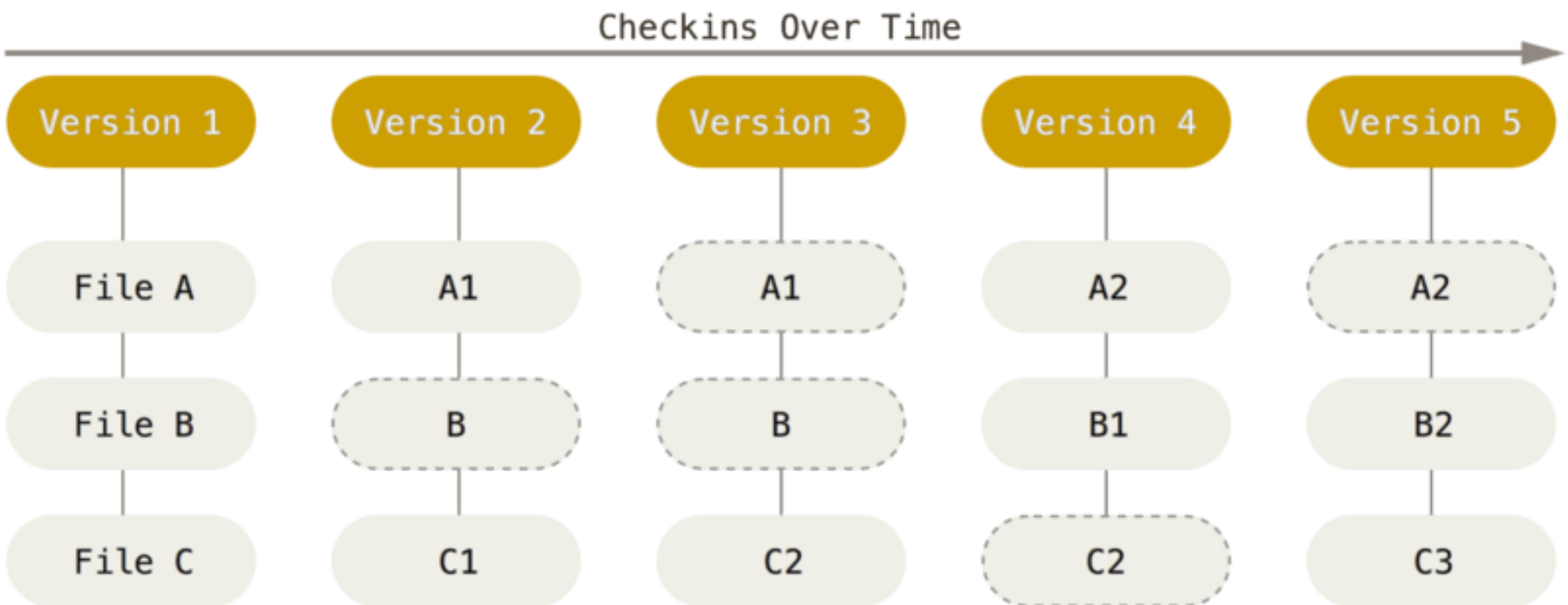
- Who am I?
- What is DaSL?
- Who are you?
  - Name, pronouns, group you work in
  - What brought you here?

Goals of the workshop

- Understand the concept of branching and merging for collaborative work.
- Create your own branch for independent work from a collaborative repository, and use a “pull request” on GitHub to receive feedback before merging.
- Create your own branch for independent work on your private repository locally, and merging your work when your independent work is complete.

Review: Git data model

You can save the state of your repository by making a **commit**: Git will save the repository's **directory tree**, a link to the previous commit, and metadata.



Branching and Merging

Linear:

```
o <-- o <-- o <-- o
```

Branching:

```
o <-- o <-- o <-- o
            ^
             \
              --- o <-- o
```

Branching and Merging:

```
o <-- o <-- o <-- o <---- o
            ^            /
             \          v
              --- o <-- o
```

Branching: when branching commit paths are created.

Merging: when two branches are integrated together. This sometimes require careful communication, and this is done in GitHub via a **“pull request”.**

Set up

1. Login to your GitHub account: https://github.com/login

2. Create a Replit account and “fork” this project:
   https://replit.com/@clo22/CollaborativeGitGitHubDaSl

4. In your Replit shell,

```
sh setup.sh
```

You will be asked how you want to log in, and pick the following:

```
Link for authentication: https://github.com/login/device
? What account do you want to log into? GitHub.com
? What is your preferred protocol for Git operations? HTTPS
? Authenticate Git with your GitHub credentials? Yes
? How would you like to authenticate GitHub CLI? Login with a web browser
```

You will be given a code, and you will provide that code to GitHub via https://github.com/login/device.

Accessing our shared repository

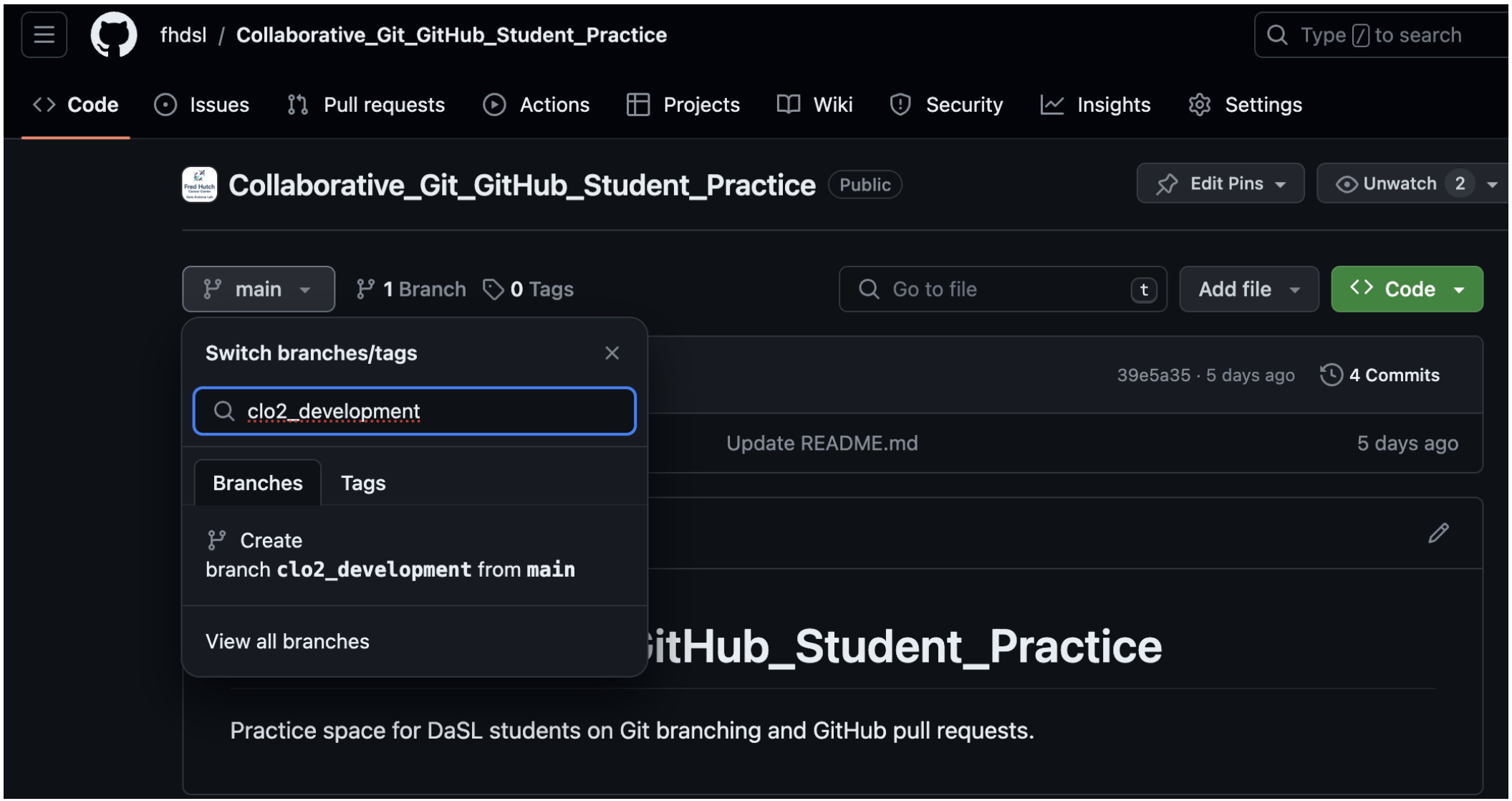
1. In Replit shell,

```
git clone https://github.com/fhdsl/Collaborative_Git_GitHub_Student_Practice.git

cd Collaborative_Git_GitHub_Student_Practice/
```

2. Open up https://github.com/fhdsl/Collaborative_Git_GitHub_Student_Practice in browser.

Creating a branch on the remote



Making changes to this new branch locally

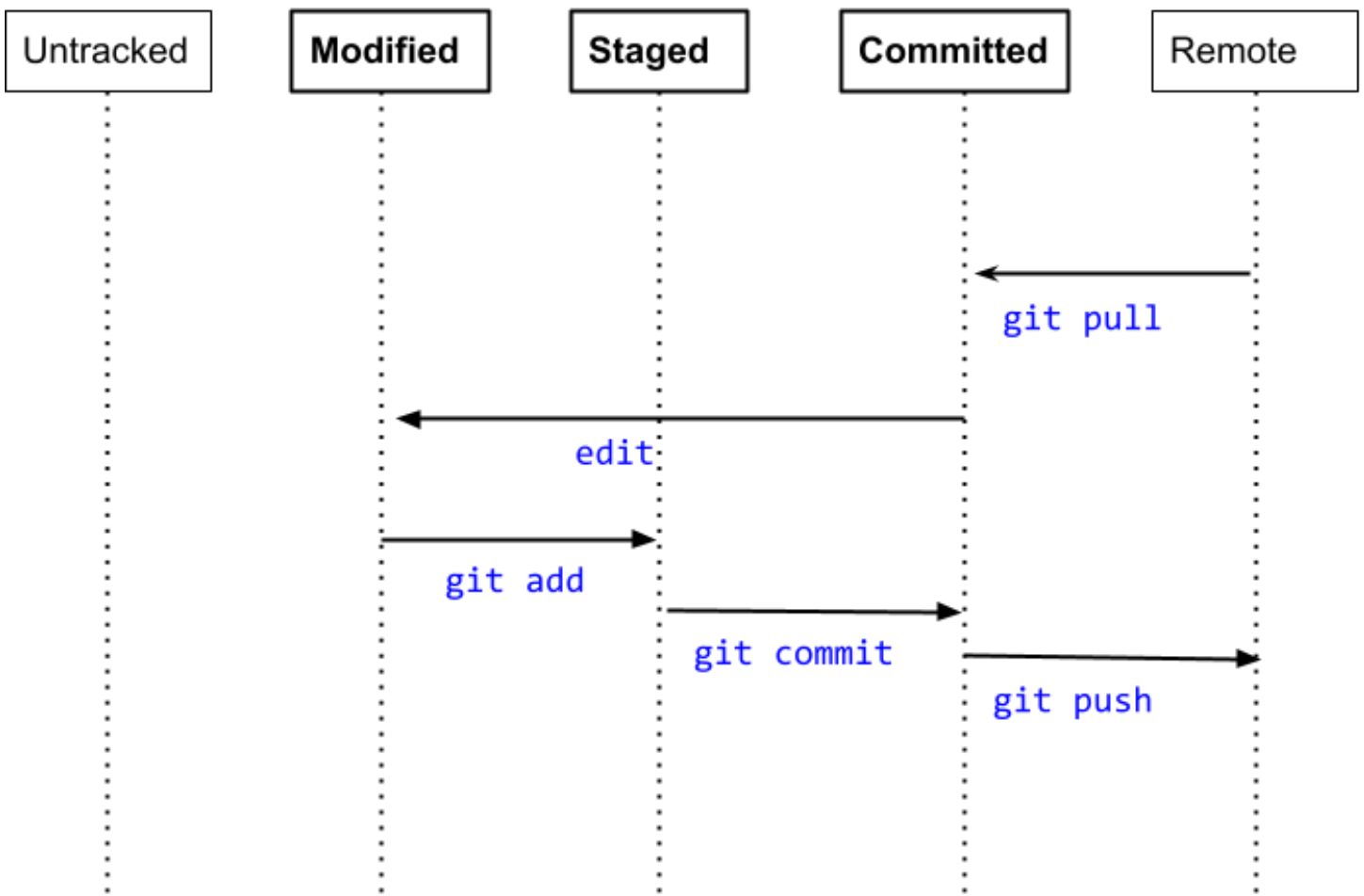
The branch `clo2_development` is created on the remote, but it hasn't been updated locally. We run `git pull` locally to update it and switch to that branch via `git checkout`.

```
% git pull
From https://github.com/fhdsl/S2_Collaborative_Git_GitHub_Student_Practice
 * [new branch]      clo2_development -> origin/clo2_development
Already up to date.

% git checkout clo2_development
Branch 'clo2_development' set up to track remote branch 'clo2_development' from 'origin'.
Switched to a new branch 'clo2_development'
```

We can use `git checkout main` to look switch back to our main branch. We can also use `git branch` to see the branches on a repository.

Review: State of a Git repository, with remote



Making changes to new branch

Create a file that is unique to you.

```
% touch clo2.txt
% echo "hello" > clo2.txt
% git add clo2.txt
% git commit -m "Created clo2.txt"
% git push
```

When you have pushed changes to the branch, you will see an option to *“Compare & pull request”*. Click on it.

clo2_development had recent pushes 2 seconds ago

Compare & pull request

clo2_developme... | 2 Branches | 0 Tags | Go to file t | Add file | Code

This branch is up to date with `main`.

Contribute

clo2fh Create newFile 413b490 · 5 days ago 2 Commits

README.md	Initial commit	5 days ago
newFile	Create newFile	5 days ago

README

S2_Collaborative_Git_GitHub_Student_Practice

Practice space for DaSL students on Git branching and GitHub pull requests.

Pull request model

A **pull request** is a way to propose changes from a branch before it is merged back into the main repository.

This is commonly used in collaborative work in which a branch needs to be approved by other members on the team before it is integrated into the main project.

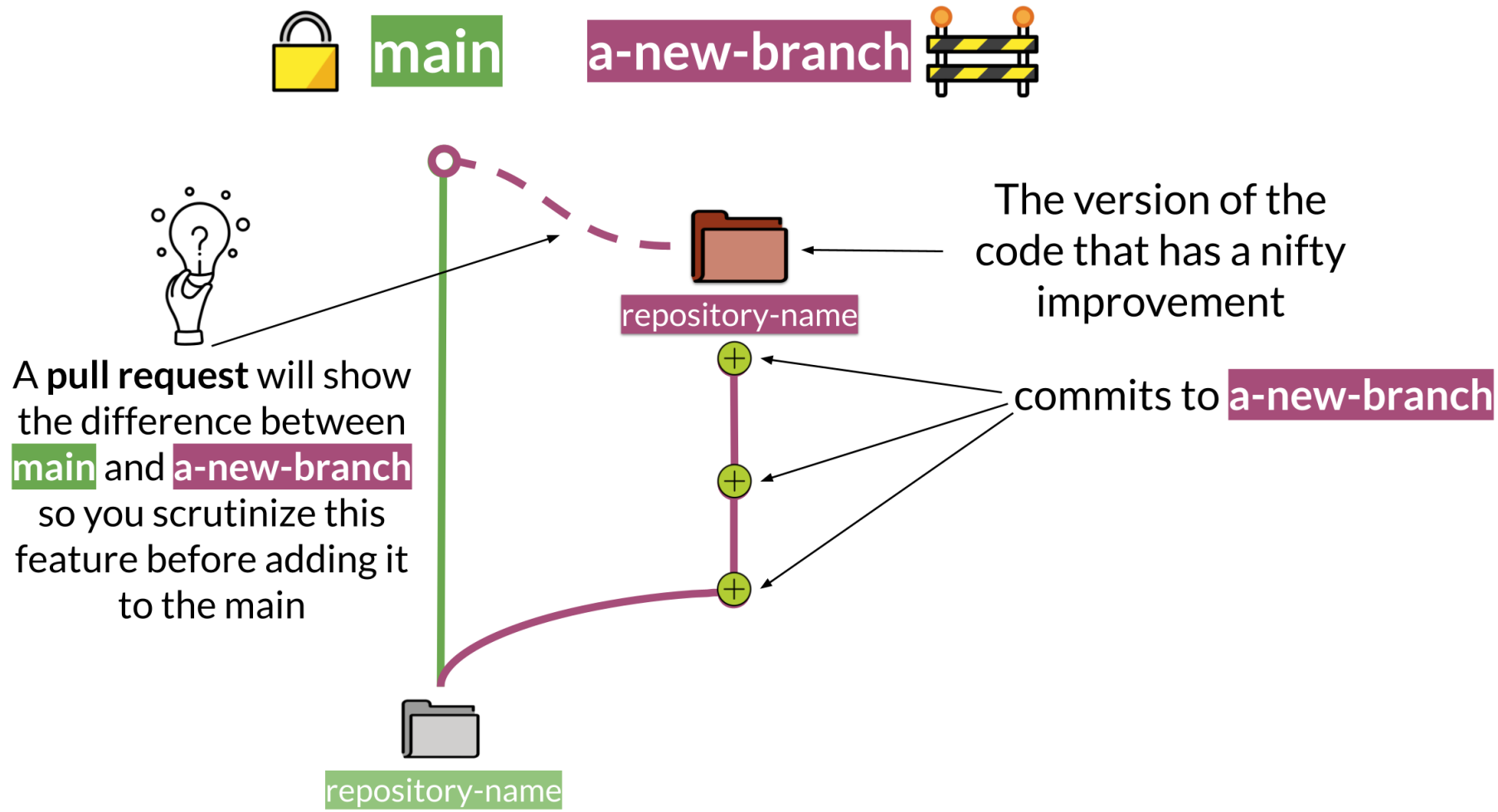
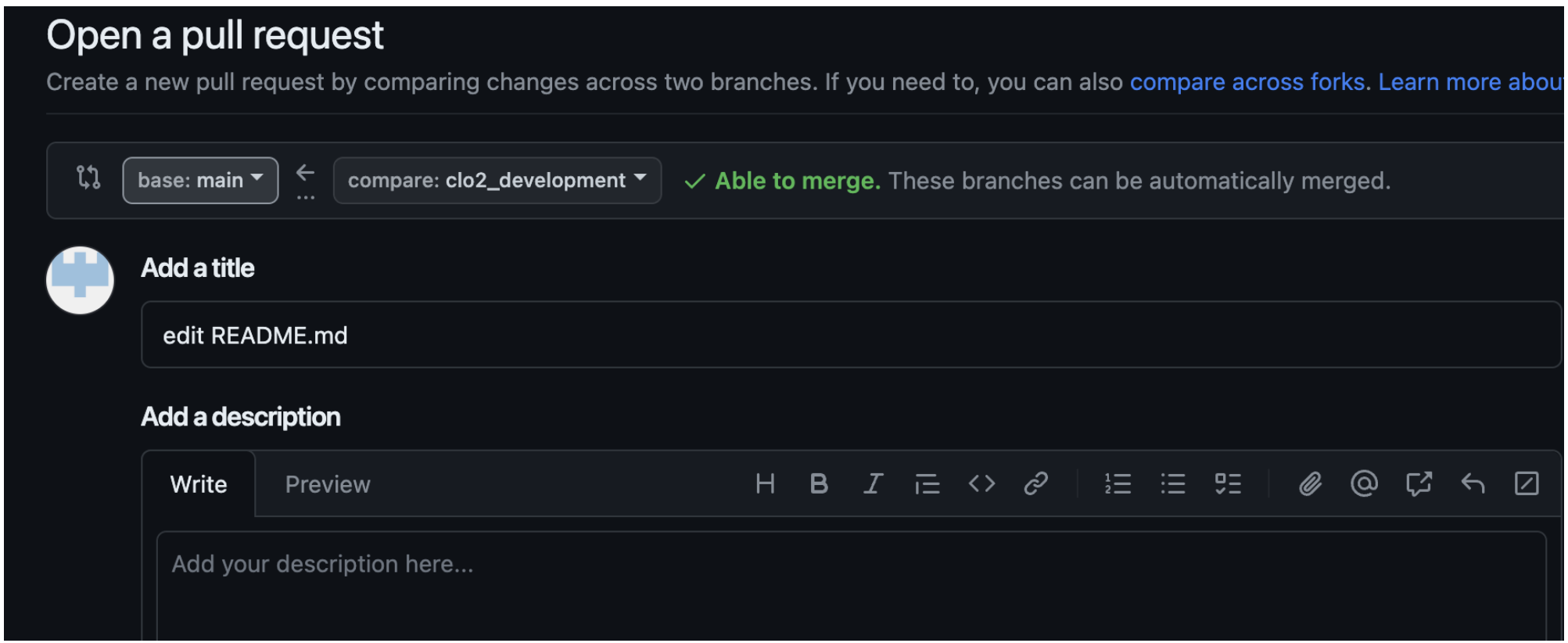


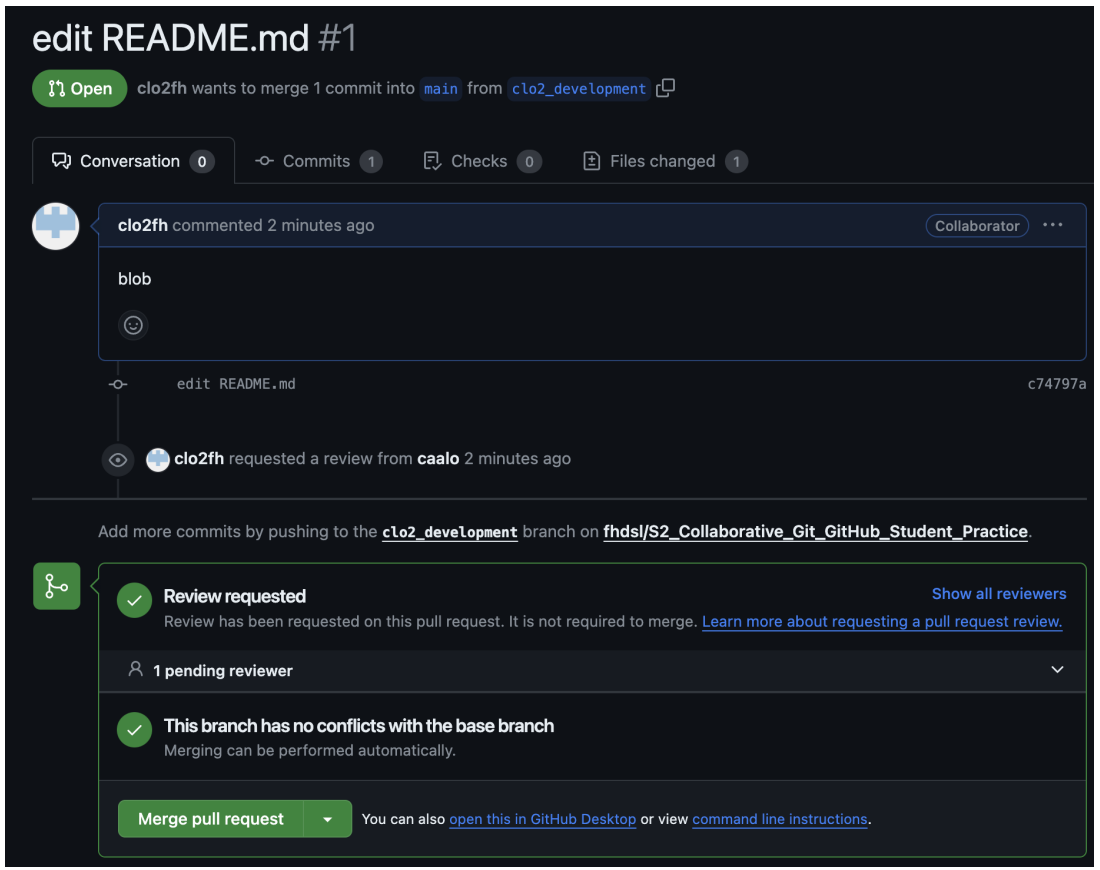
Image by Candace Savonen

Creating a pull request

You will see that you are trying to merge `clo2_development` into `main` on the remote. It also requires you to write a description of what you did on your branch.



Creating a pull request



Add your partner's GitHub username as your reviewer, and have them make comments/create a code review about it!!

Make additional commits based on their comments.

Guidelines on pull request discussions

For writers:

- it provides context of the code changes you made.
- it asks for explicit feedback of what kind of feedback is needed.
- it is a a small and modular change that can be discussed.

For reviewers:

- Do the proposed changes answer the solve the problem? Can you test it out in the working branch?
- Is the code clear and readable?
- Is the code efficient with computational resources?
- Does the code stick to the style and conventions of this project?

Click *“Merge pull request”* to finish!

A Pull Request with conflicts

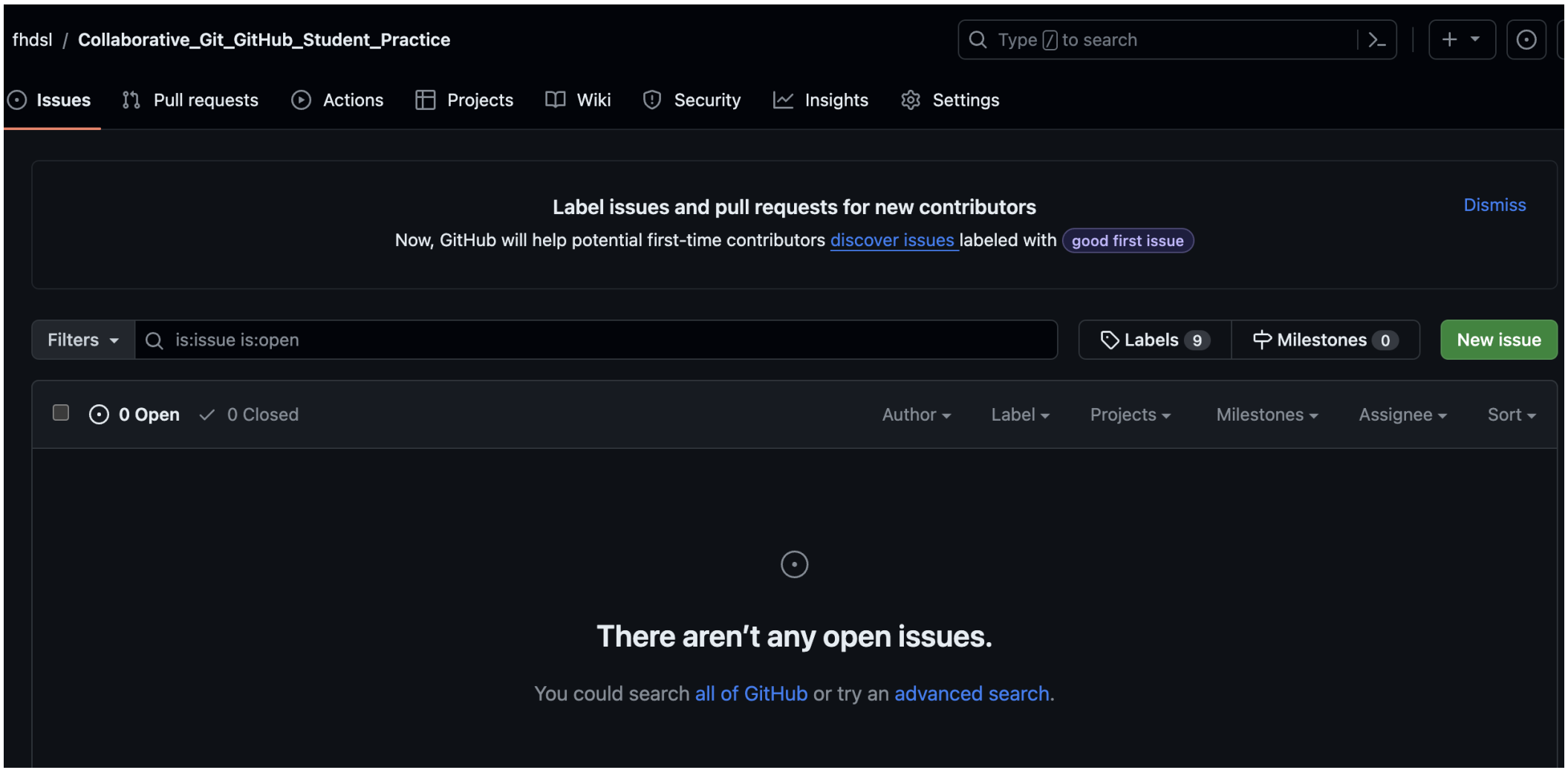
- Everyone create a new branch
- I'll modify `README.md` on the main branch
- You make changes to `README.md` on your branch, and make a Pull Request
- What to do with a conflict?

GitHub issues

GitHub issues are a way for people to give feedback on your repository:

- You publish a piece of software on GitHub, and other users try it out.
- They are confused about how to run your code, so they return to the GitHub repository and post a GitHub issue documenting their error.
- You can then create a branch from this issue, make changes to your code to resolve the error, then use the pull request model to merge it back to the main branch!

Try creating a GitHub issue



Getting started with branches

Let's create a local repository to practice branching:

```
% mkdir sandbox
% cd sandbox
% git init
% touch README
% git add README
% git commit -m "Added README"
```

Let's look at the branches in this repository:

```
% git branch
* main
```

The star `*` shows which branch we are looking at.

We are on the `main` branch in this repository, as expected.

Ways to look at a branch

Let's create a new branch:

```
% git branch development
% git branch
  development
* main
```

Another way to look at which branch we are on is via `git log`:

```
% git log
commit 657fcbea5a023041d359a8f1fcfc9fbf7e64f68e (HEAD -> main, development)
Merge: 875d774 413b490
Author: Your Name <you@example.com>
Date:   Wed Dec 6 23:47:39 2023 +0000

   Initial commit
```

The HEAD **pointer** tells us "What am I looking at?" in our local file system.

Committing to the development branch

```
% git checkout development
Switched to branch 'development'
% echo "Additional README info" >> README.md
% git add README.md
% git commit -m "updated README"
```

Let's look at our `git log`:

```
% git log
commit 260c8099f0ea82199805325a6fbe26bfc3cbd1aa (HEAD -> development)
Author: Your Name <you@example.com>
Date:   Thu Dec 7 00:14:02 2023 +0000

    updated README

commit 657fcbea5a023041d359a8f1fcfc9fbf7e64f68e (main)
Merge: 875d774 413b490
Author: Your Name <you@example.com>
Date:   Wed Dec 6 23:47:39 2023 +0000

    Initial commit
```

Now, our branch `development` is ahead of the `main` branch by one commit. We can toggle between two branches via `git checkout` as before.

Merging

All you have to do is checkout the branch you wish to merge into and then run `git merge [branchName]` on the branch of interest:

```
% git checkout main
% git merge development
Merge branch 'main' of https://github.com/fhdsl/Collaborative_Git_GitHub_Student_Practice
Updating 657fcbe..260c809
Fast-forward
% git log
commit 260c8099f0ea82199805325a6fbe26bfc3cbd1aa (HEAD -> main, development)
Author: Your Name <you@example.com>
Date:   Thu Dec 7 00:14:02 2023 +0000

    updated README

commit 657fcbea5a023041d359a8f1fcfc9fbf7e64f68e
Merge: 875d774 413b490
Author: Your Name <you@example.com>
Date:   Wed Dec 6 23:47:39 2023 +0000

    Initial commit
```

Appendix: References

- ProGit: We covered chapter 3 in this workshop.
- DangItGit: Excellent starting point for common Git problems.
- MIT's Git Seminar: A more computer science explanation of how Git works.
- Explain Shell: Access Shell and Git manual and help pages in an easy-to-read way.