
Middlebury

# CSCI 201: Data Structures

## Fall 2024

---

# Lecture 12T: Software Development I (Repositories)

# Goals for today:

**Motivation:** is there a good way to save different versions of our codes?

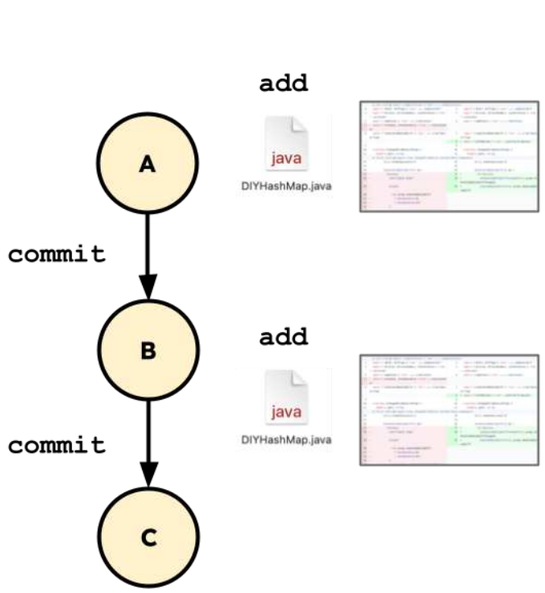

- Use `git` to keep track of different version of your codes.
- Use `GitHub` as a central (**remote**) location for your code repository.
- Save code changes using the `add` (**stage**) and `commit` steps.
- **Synchronize** commits to/from the remote repository using `push` and `pull`.
- Practice indexing cells in a **grid** (useful for Homework 10).

Useful for electives, personal projects, and in industry.

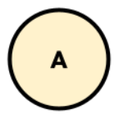
# `git` and `GitHub` in a nutshell (`git` ≠ `GitHub`).

- `git` is a *version control system*.
- `git` stores your code as a series of **snapshots** represented as a **graph**.
- Making a `commit` takes a snapshot of your code (creating a new node in the graph).
- `GitHub` is a *repository hosting* service: a place to store a central graph.
- When you `clone` a repository, you're copying the graph.
- Your local repository and the remote repository may not always be in sync.

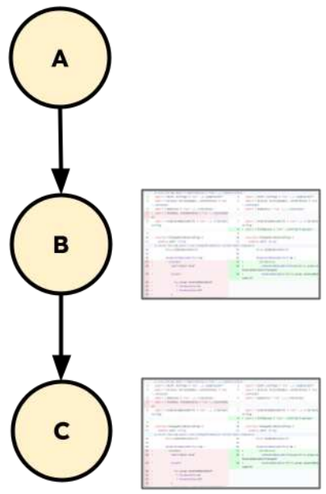
**On your computer (or other development machine):**




**On `GitHub`:**

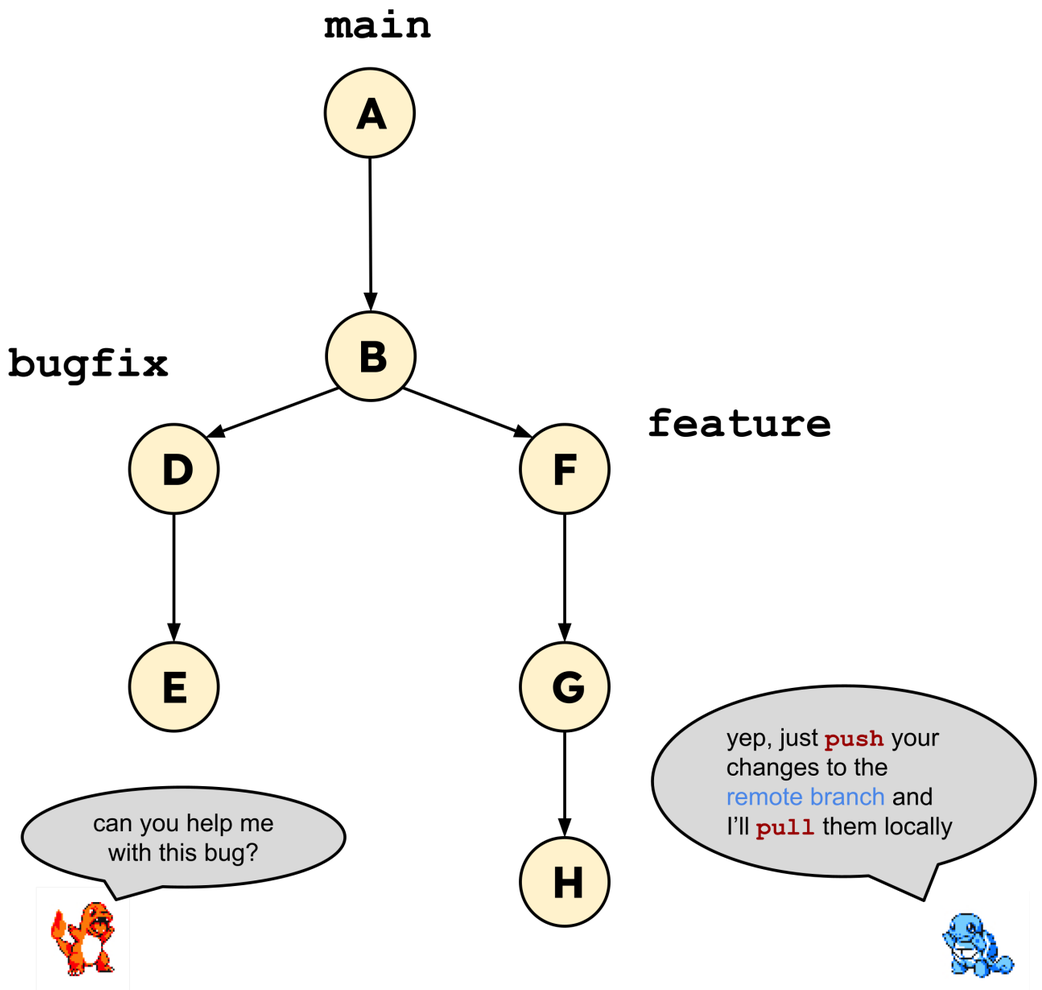



**On `GitHub` (after `push`):**




**Main concepts: `add`, `commit`, `push`**

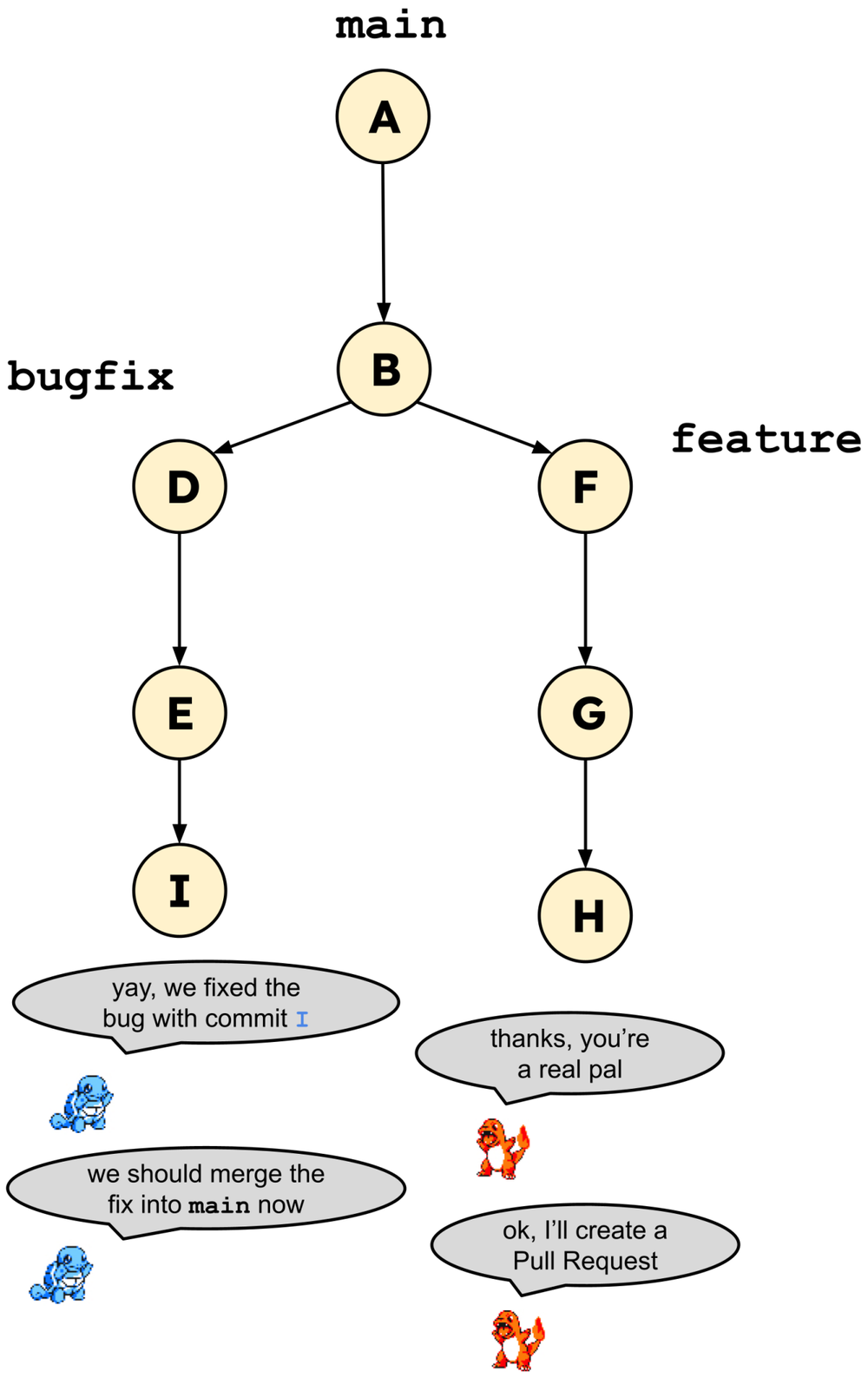
Usually, you'll work in a *branch*. If you're working alone, you can just work in a single (`main`) branch.

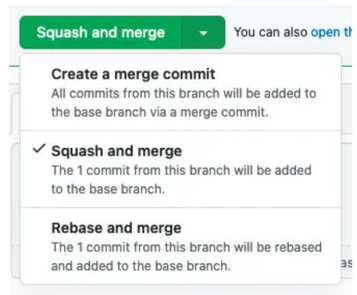


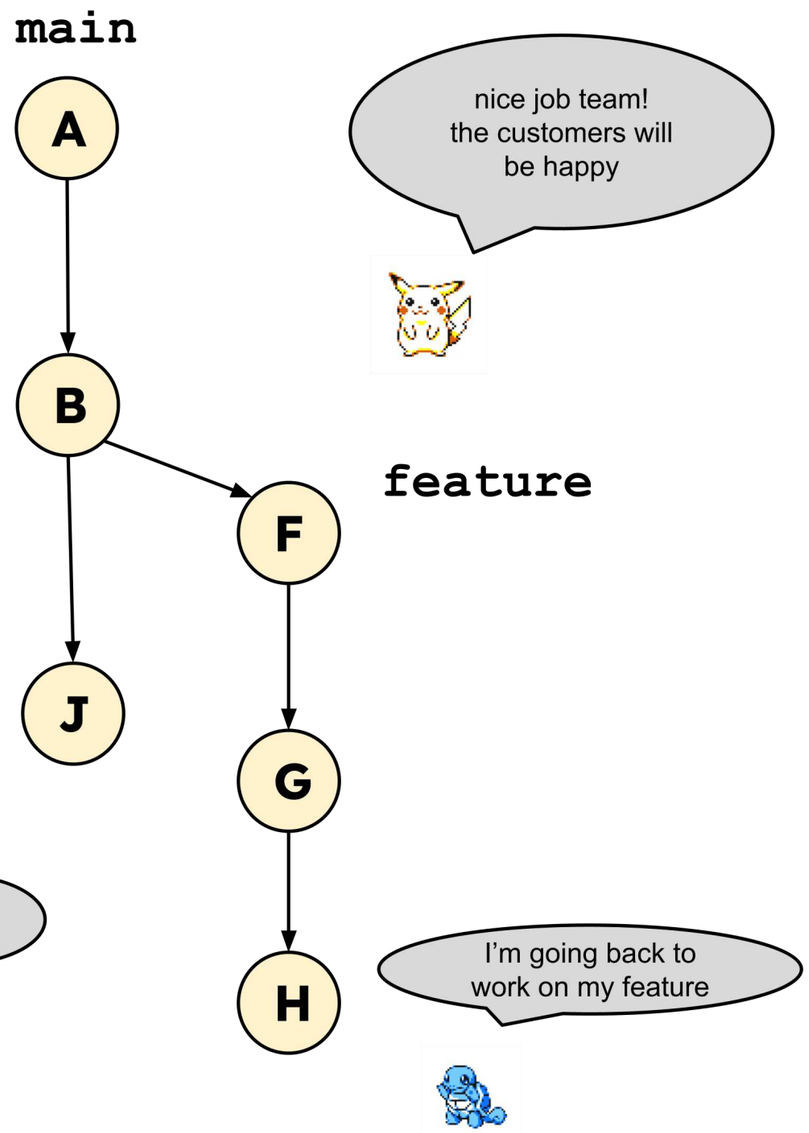

`push` your commits to the remote repository

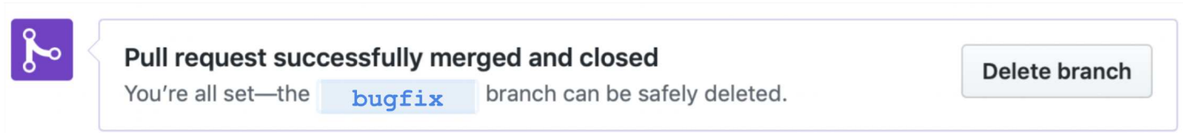
`pull` commits from the remote repository

# Working on different branches in a team.



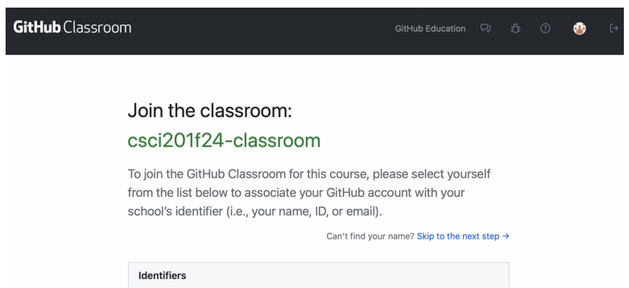



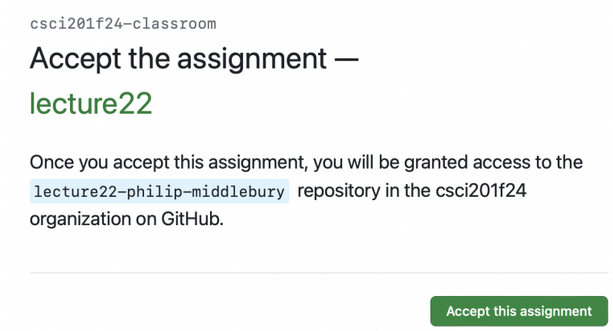



# Now, we'll practice with `git` and `GitHub`.

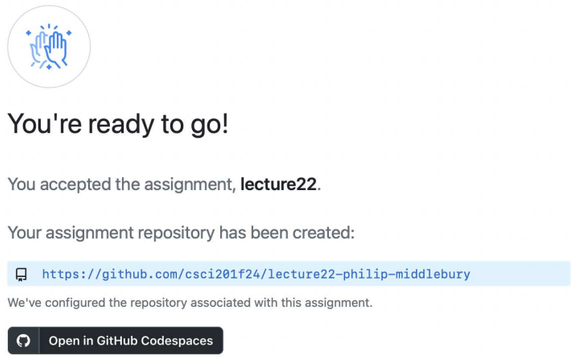
1. If you do not already have an account on `GitHub`, create one here: https://github.com/signup.
2. Then click on the `code` link on the course webpage: https://classroom.github.com/a/bBsdeyav.



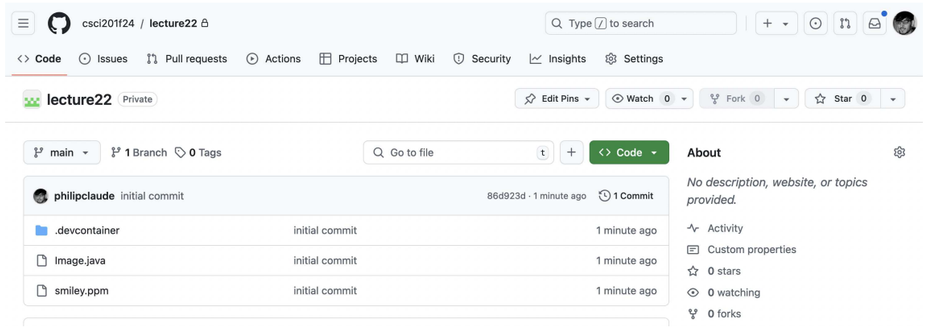

Find your name and join the classroom.




Accept the assignment.



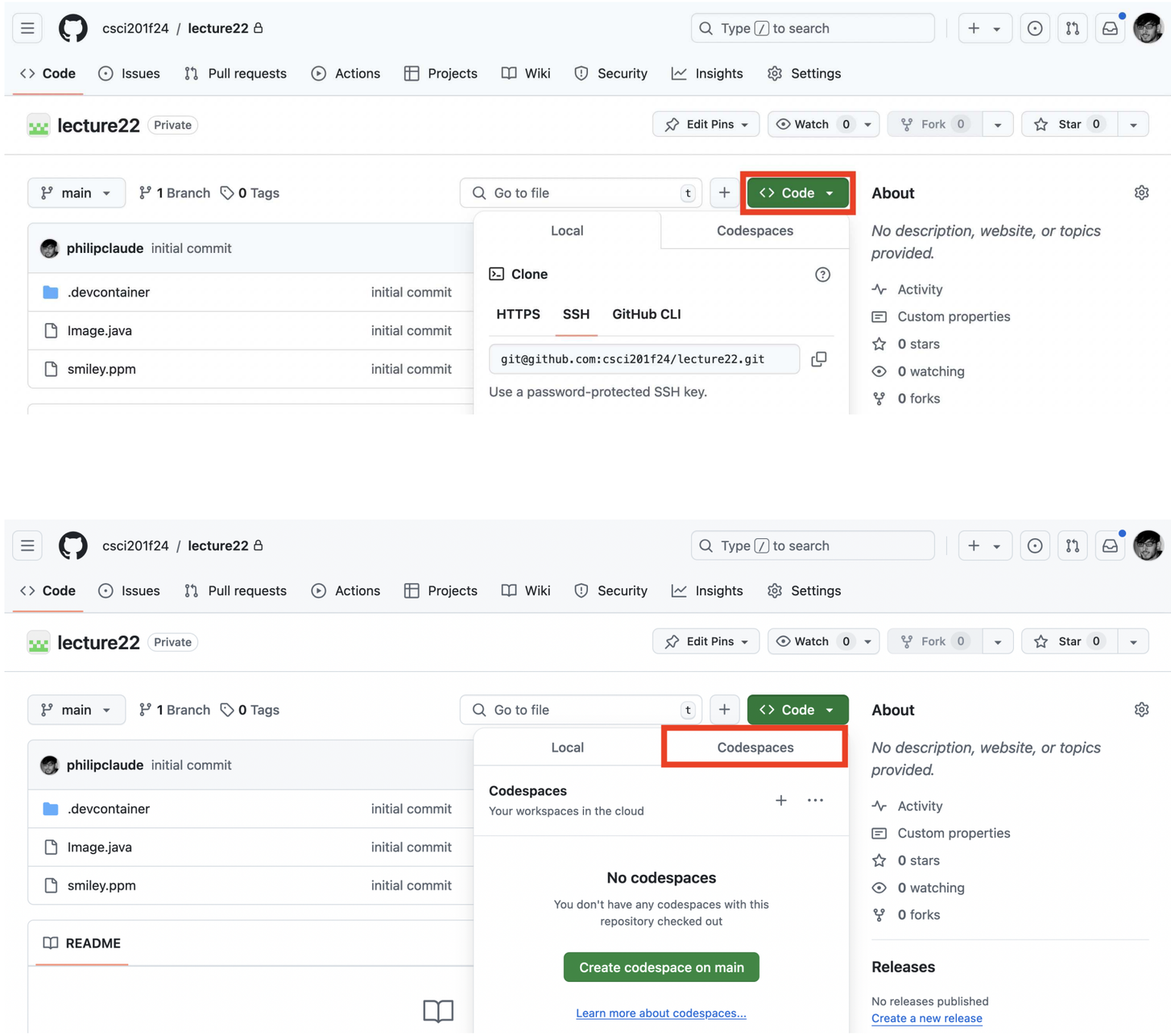

Click on the link to open the repository.



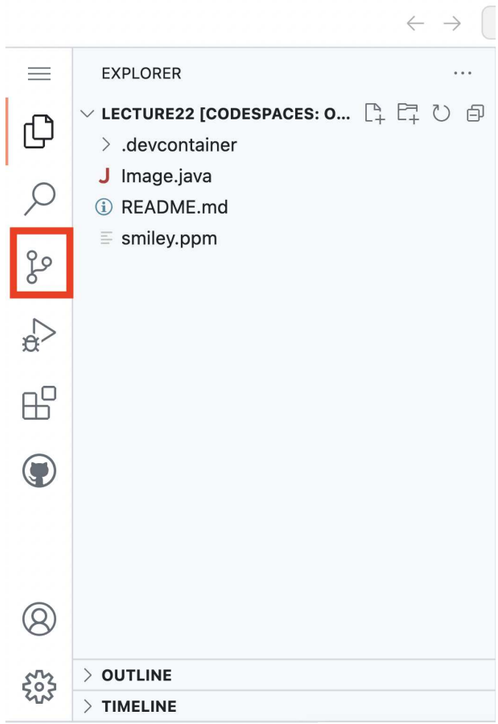

You should then see this.

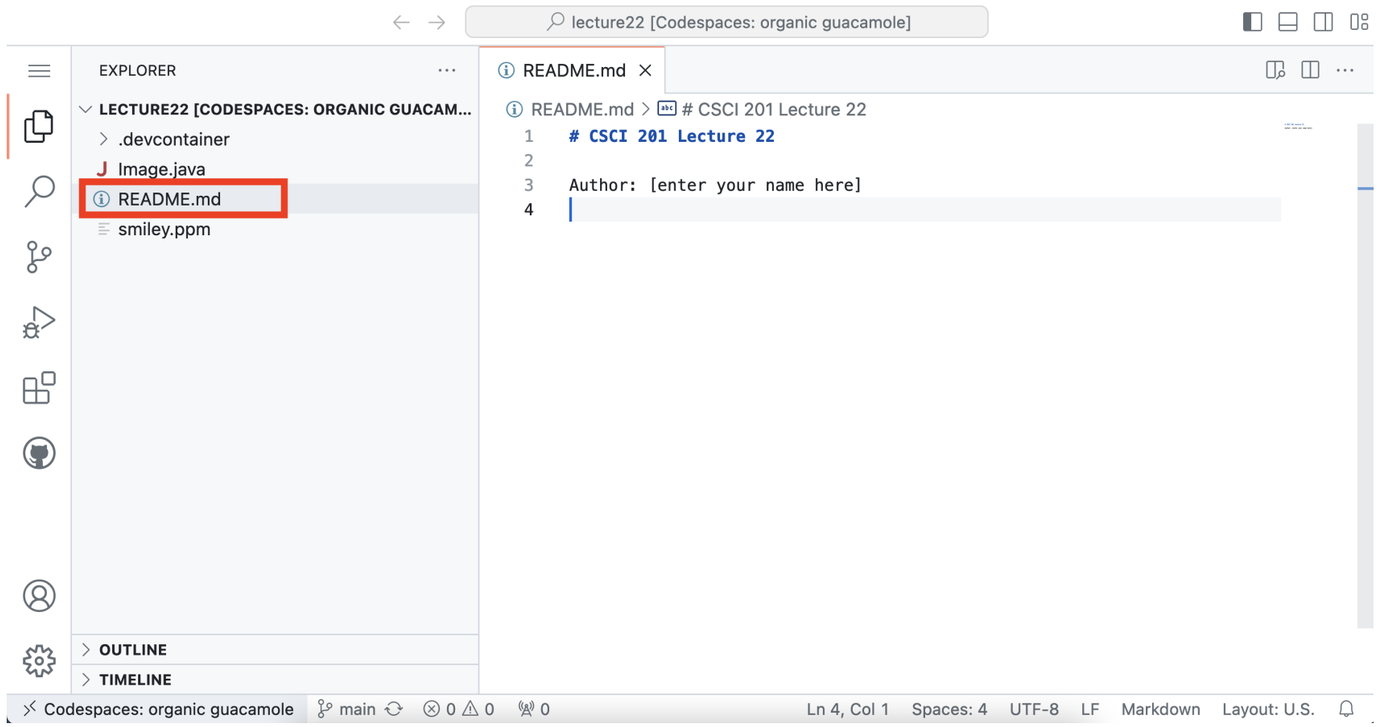
# Our "local" computer will be a virtual machine in the cloud.

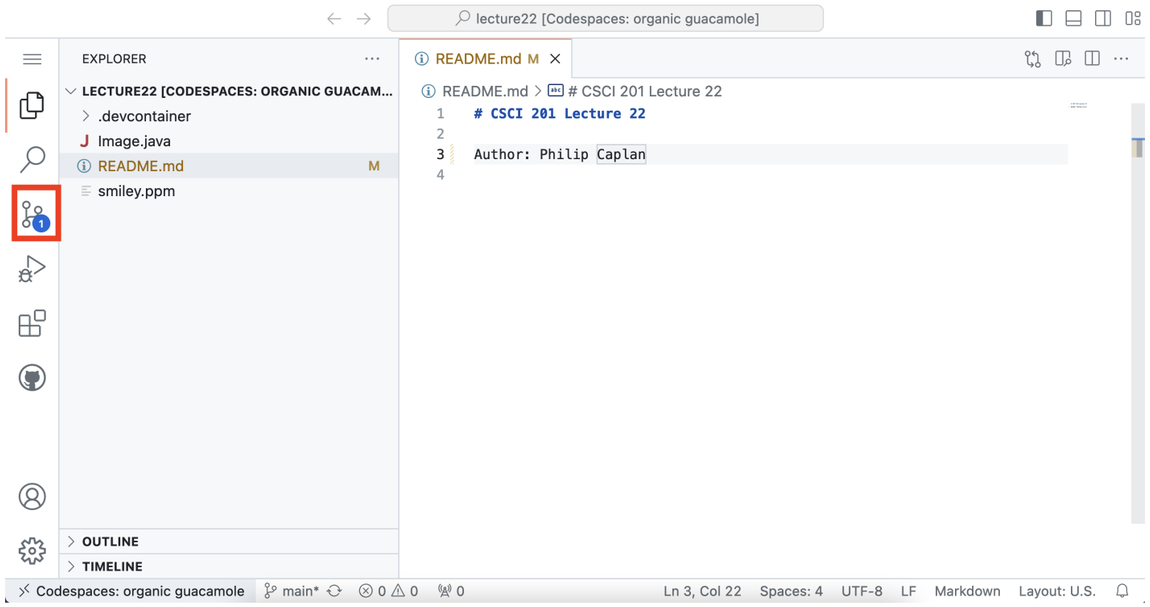
We'll work on our code in a `GitHub` **Codespace** (which will have VS Code integrated).



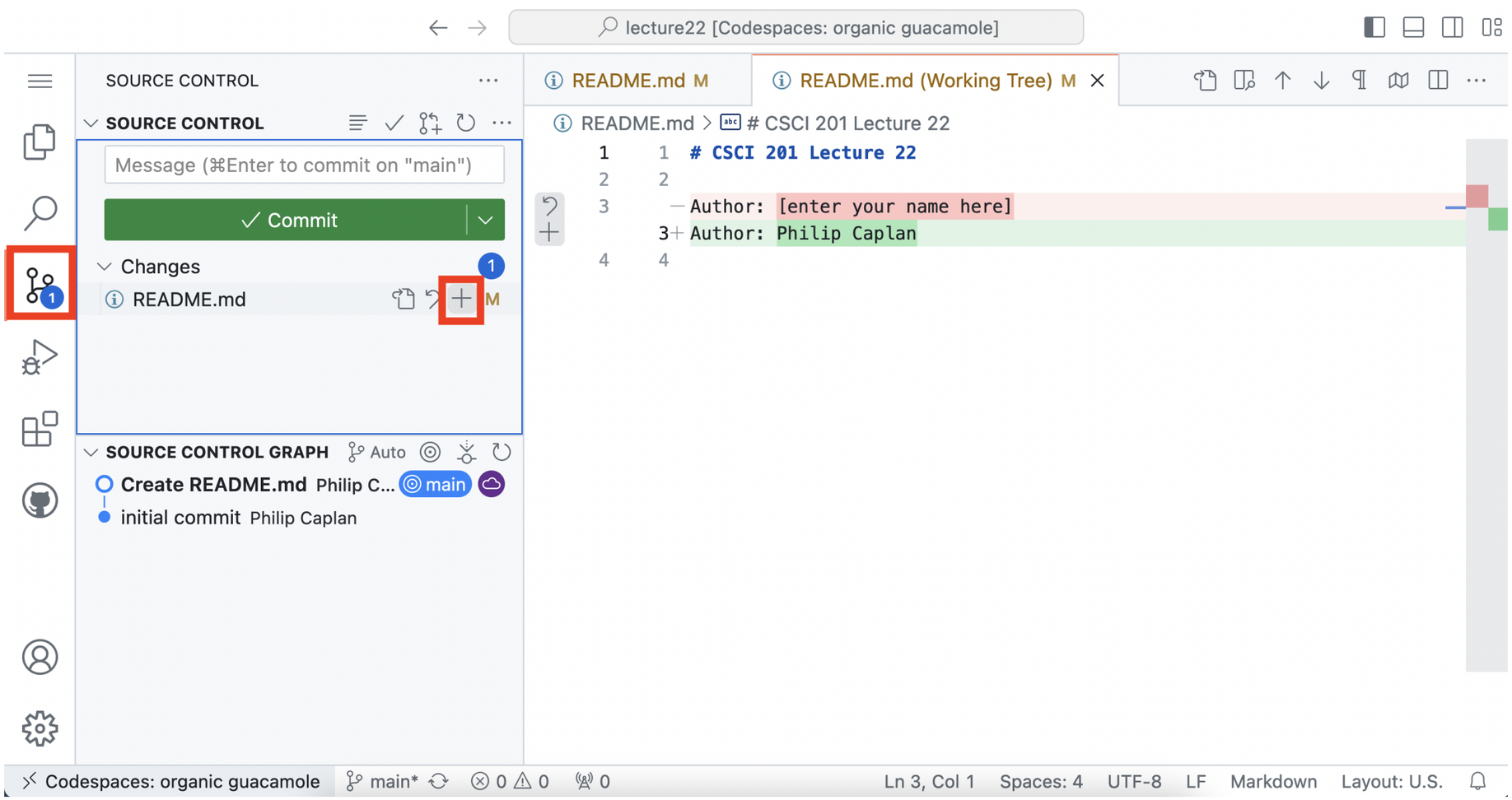
# The *Source Control* button can be used to do `git` commands.



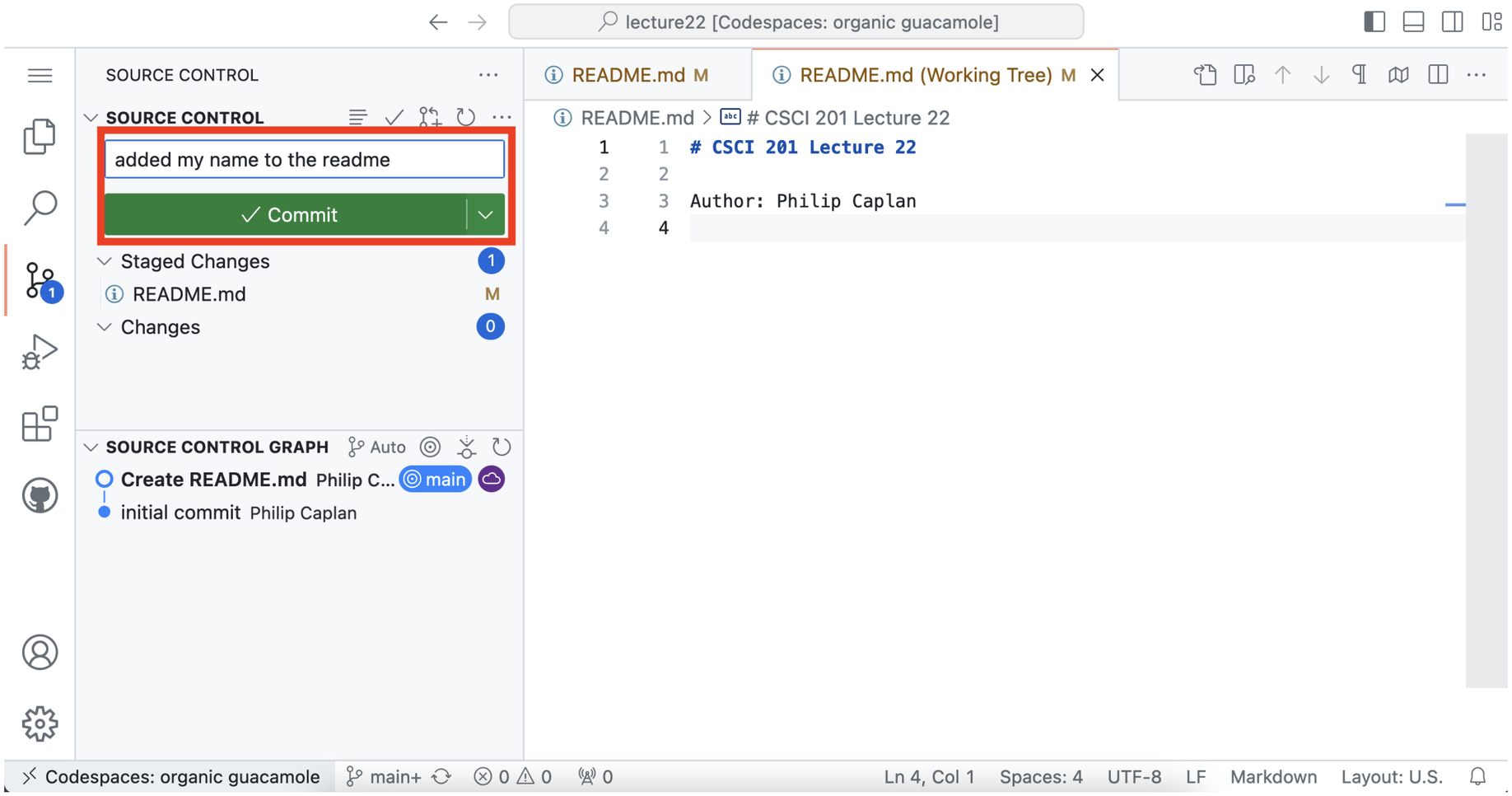



# Inspect the changes by clicking on a file in the "Changes" section.

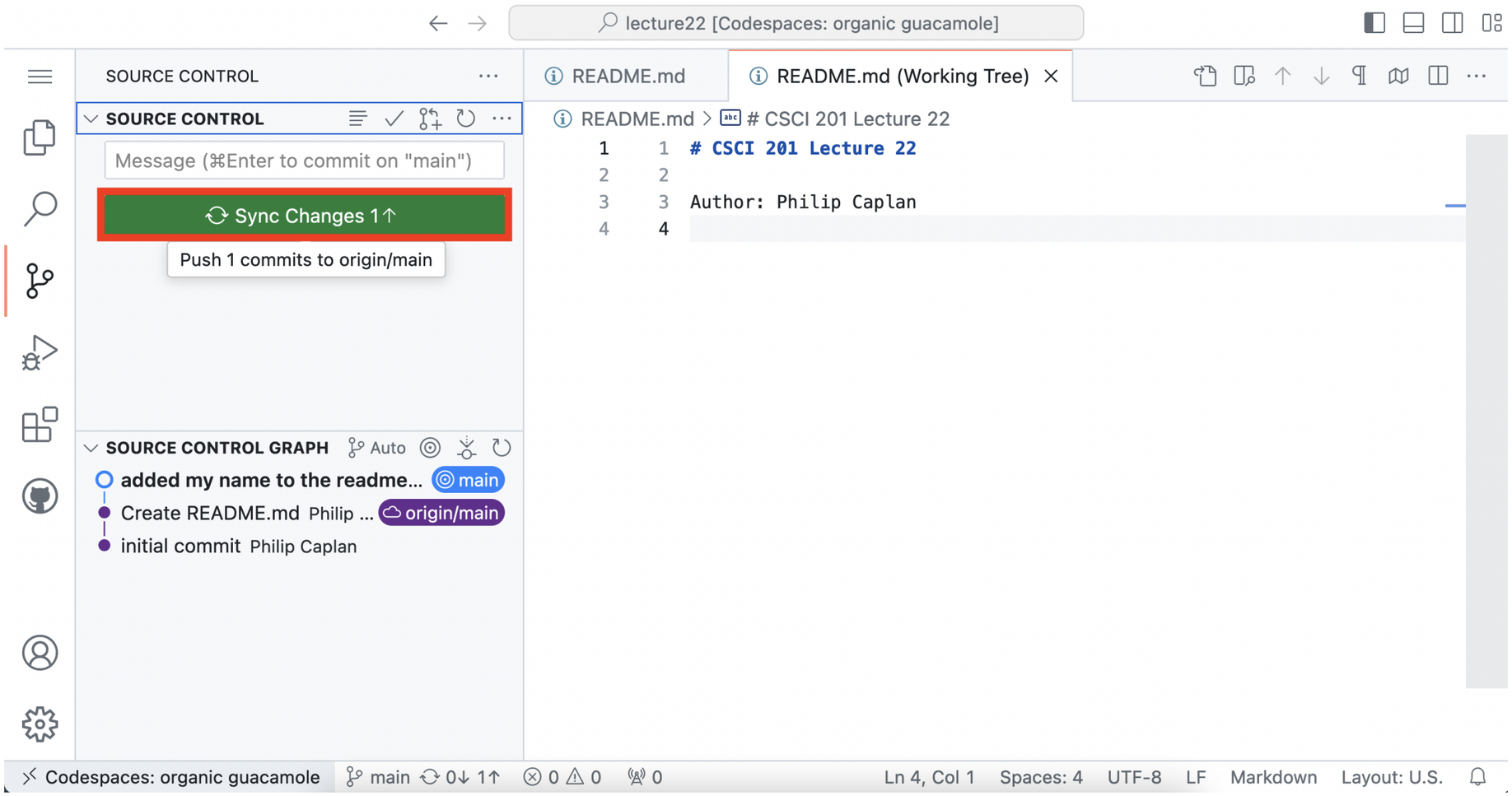



Then click on the + button to "add" (or "stage") your changes.

# Add a message (information about the changes) and then click the "Commit" button.

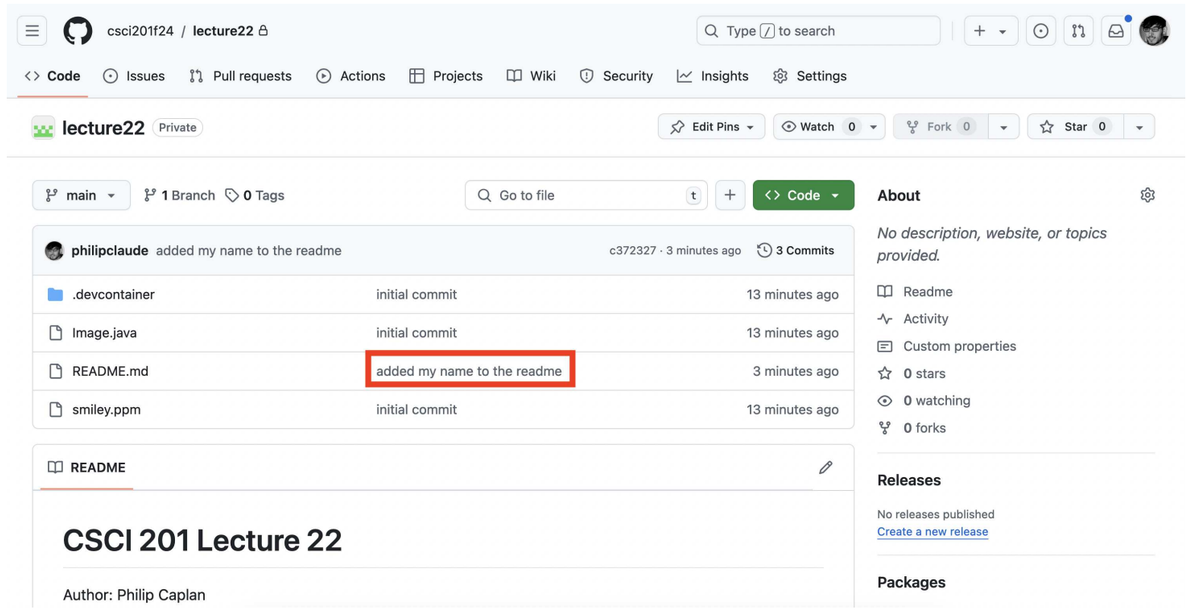


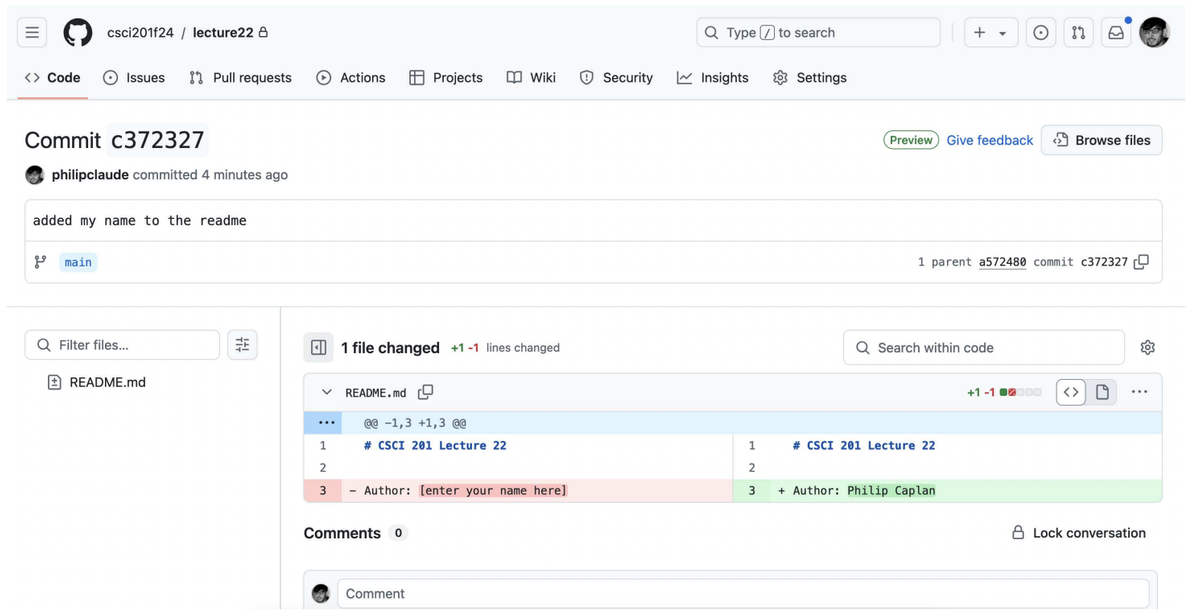
# Finally, `push` the changes to the remote repository (on `GitHub`) by clicking on "Sync Changes".



# Now the `GitHub` repository has the commit we just made.

We can also click on the commit message to get more info about the changes.



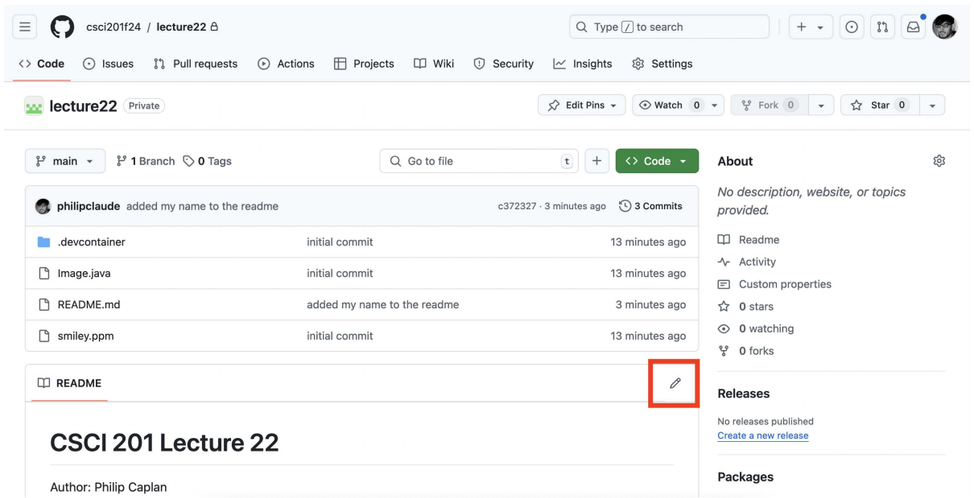



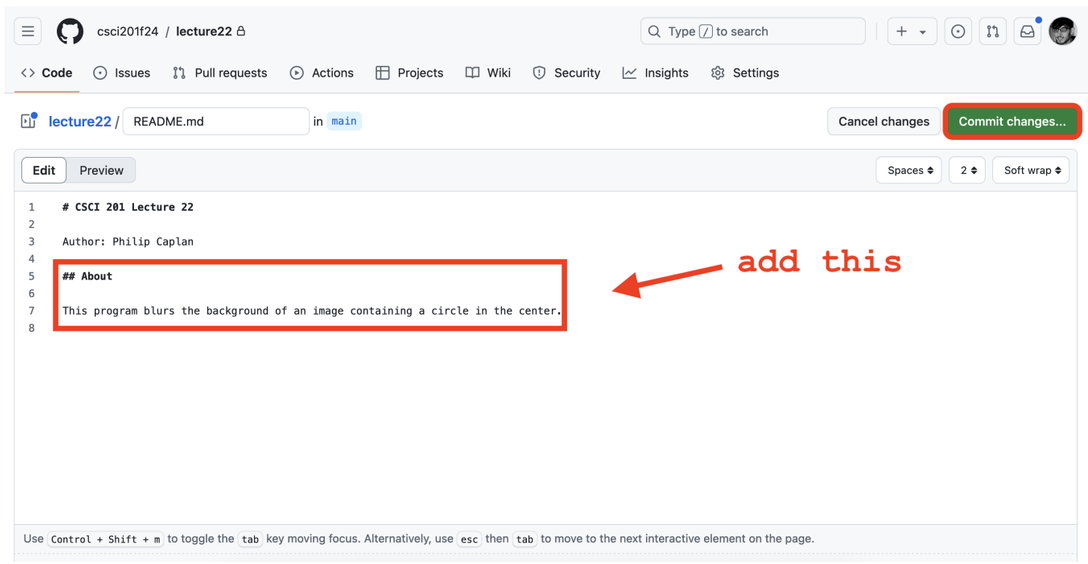
# Our problem for today: a bit of image processing (to practice indexing grid cells).

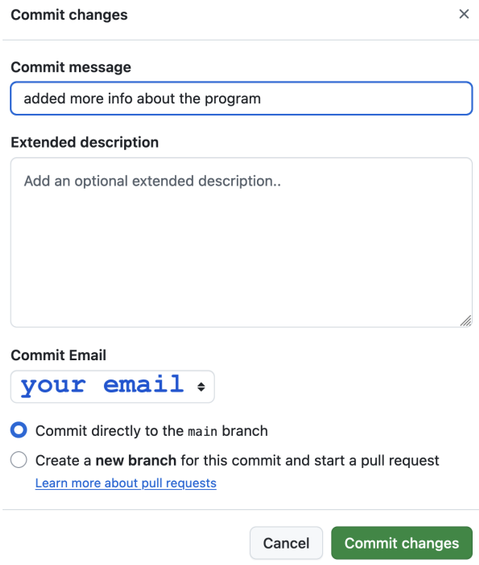
- Smiley wants to blur the background of each frame (in a video chat) so it doesn't look like a ski hill.
- We'll help Smiley by writing a program to blur the background pixels.

# But first, let's practice with `committing` changes and `pulling`.

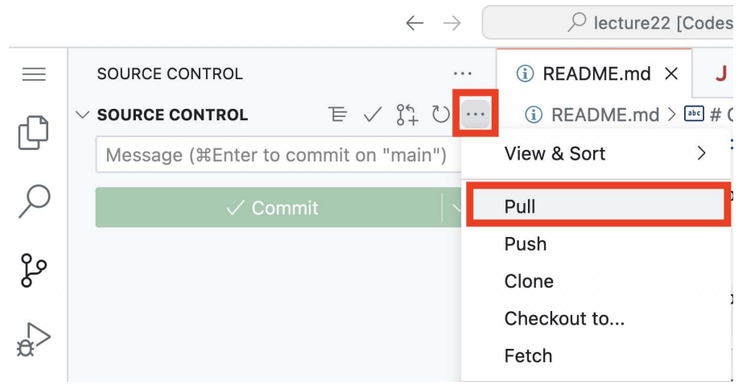
We can also do this directly in `GitHub`!



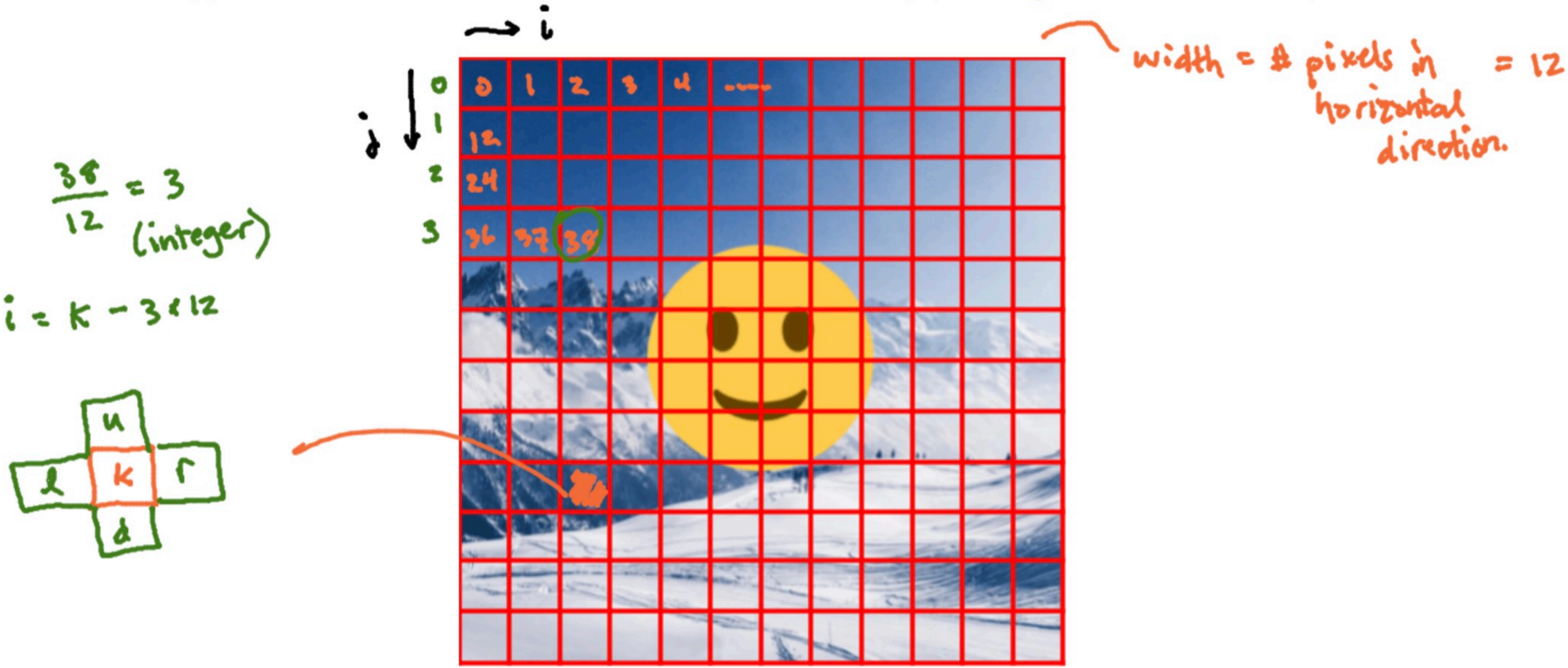
# Now retrieve the changes in your Codespace.





Now we know how to `add` (stage), `commit`, `push` and `pull`.

Here, pixels are stored in a one-dimensional array (`ArrayList<Pixel>`).



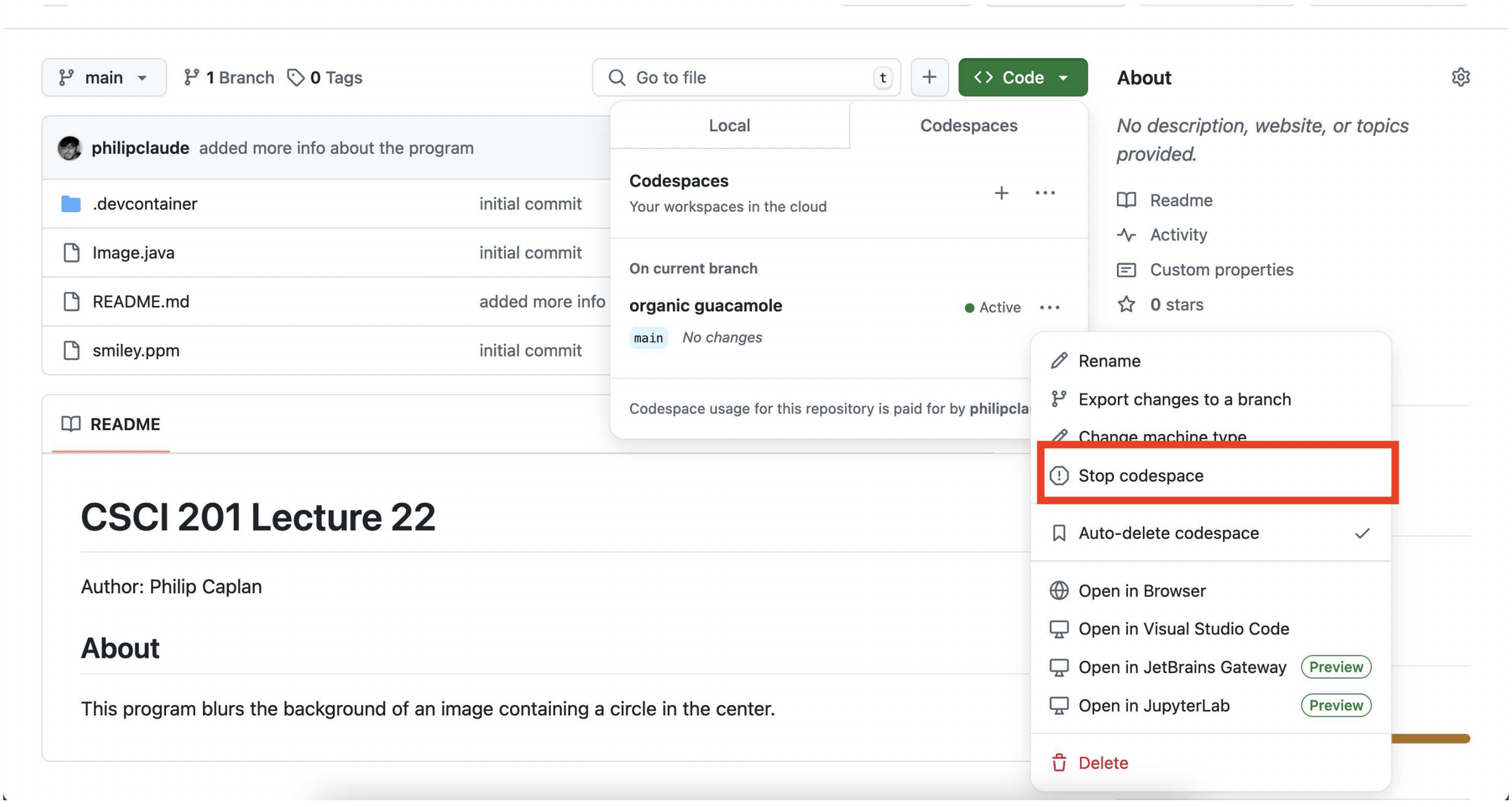

- Every pixel outside of the circle will be set to the average of the surrounding pixels (right, left, above, below).
- To determine if a pixel is outside the smiley face circle, we need the row and column indices of the pixel.
- How to retrieve `i` (column) and `j` (row) indices from pixel index `k`?
- How to retrieve the surrounding pixels (right, left, above, below)?
- We will not modify pixels on the image boundary. How to detect these and skip them?

# Possible implementation of the `blurBackground` method.

```
1  public void blurBackground(int faceRadius, int numIterations) {
2    int cx = width / 2;
3    int cy = height / 2;
4
5    for (int iter = 0; iter < numIterations; iter++) {
6      for (int k = 0; k < pixels.size(); k++) {
7        int j = k / width;
8        int i = k - j * width;
9
10       if (i == 0 || i + 1 == width || j == 0 || j + 1 == height) {
11         continue;
12       }
13
14       int radiusSquared = (i - cx) * (i - cx) + (j - cy) * (j - cy);
15       if (radiusSquared < faceRadius * faceRadius) {
16         continue;
17       }
18       Pixel pixel = pixels.get(k); // or j * width + i
19       Pixel r = pixels.get(j * width + i + 1); // or k + 1
20       Pixel d = pixels.get((j + 1) * width + i); // or k + width
21       Pixel l = pixels.get(j * width + i - 1); // or k - 1
22       Pixel u = pixels.get((j - 1) * width + i); // or k - width
23       pixel.average(r, d, l, u);
24     }
25   }
26 }
```

Now, `add` (click the `+` button for `Image.java`), `commit` (adding a message), then sync (`push`).

# It's a good idea to stop your codespace.



# Additional notes:

- For more info: https://code.visualstudio.com/docs/sourcecontrol/intro-to-git
- **Homework 10** due Tuesday 12/10.
- Last Exit Ticket (12T): **install a few tools** before Thursday's class.
- Thursday we will talk about **testing**.
- Friday's lab (**Lab 10**) will have two components:
  - We will complete the **Course Response Form** in the lab period.
  - You will submit your reflection for your **Participation** grade.
- **Final Exam:** (released Tuesday 12/10 and will be due Friday 12/13)
  - **Two components:** Part 1 on Canvas and Part 2 (programming) submitted to Gradescope.
  - **Study Guide** will be posted by the end of the week.