

# Git It Together

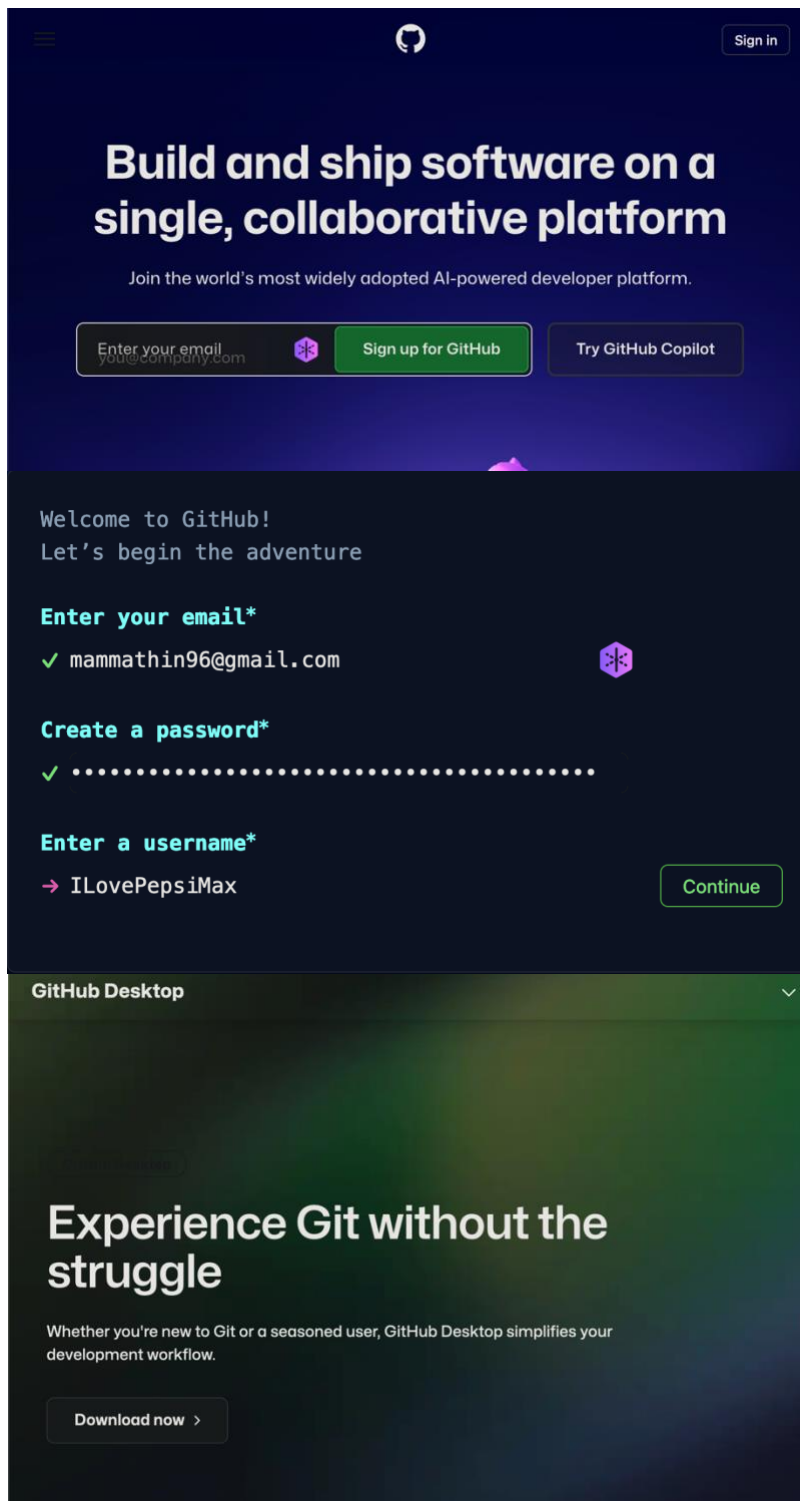
## The Sacred GitHub Handbook

### Table of Contents

Creating an account with GitHub.....	2
GitHub Desktop .....	2
Cloning a Repository .....	3
Pushing .....	5
Pulling .....	5
Branching .....	6
Creating a branch.....	6
Merging Branches .....	7
Code Reviews .....	10
Branch Strategies.....	11
Common Problems and Solutions .....	12

## Creating an account with GitHub

To create an account on GitHub, head over to [github.com](https://github.com) and locate the „Enter your email“ text box, once you have typed in your email press the signup button



Build and ship software on a single, collaborative platform

Join the world's most widely adopted AI-powered developer platform.

Enter your email  
yourcompany.com

Sign up for GitHub

Try GitHub Copilot

Sign in

Welcome to GitHub!  
Let's begin the adventure

Enter your email\*

✓ mammathin96@gmail.com

Create a password\*

✓ .....

Enter a username\*

→ ILovePepsiMax

Continue

GitHub Desktop

Experience Git without the struggle

Whether you're new to Git or a seasoned user, GitHub Desktop simplifies your development workflow.

Download now >

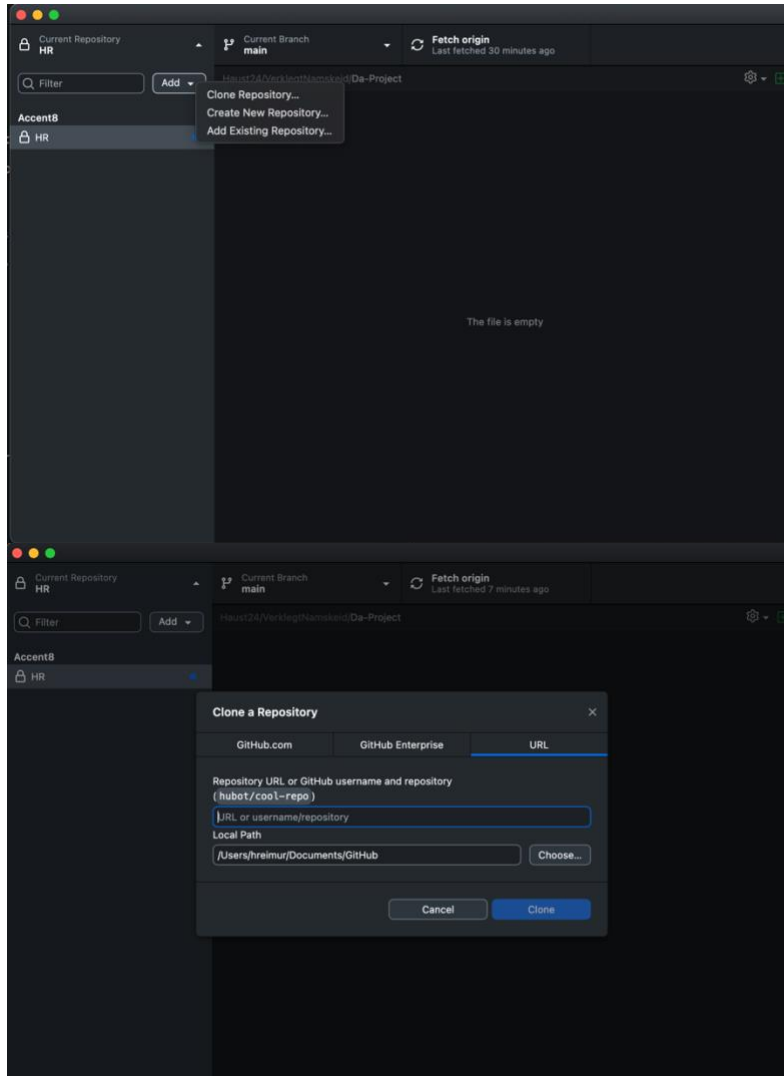
After the sign in, you will be taken to another page. On that page fill out the information they ask for

Then you do a short puzzle to make sure you are not a bot and then you are off to the races of GitHub and it's amazing features.

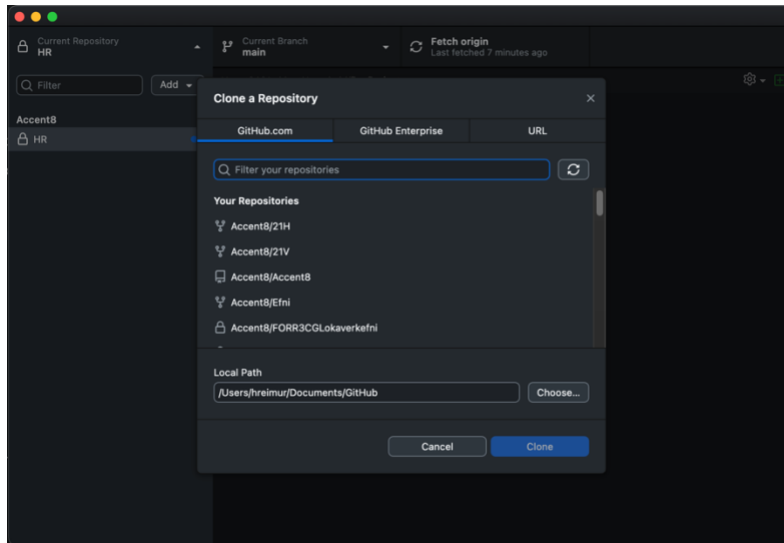
**GitHub Desktop** After doing that you head over to [desktop.github.com](https://desktop.github.com) and download GitHub desktop then just sign in with the account you just made.

## Cloning a Repository

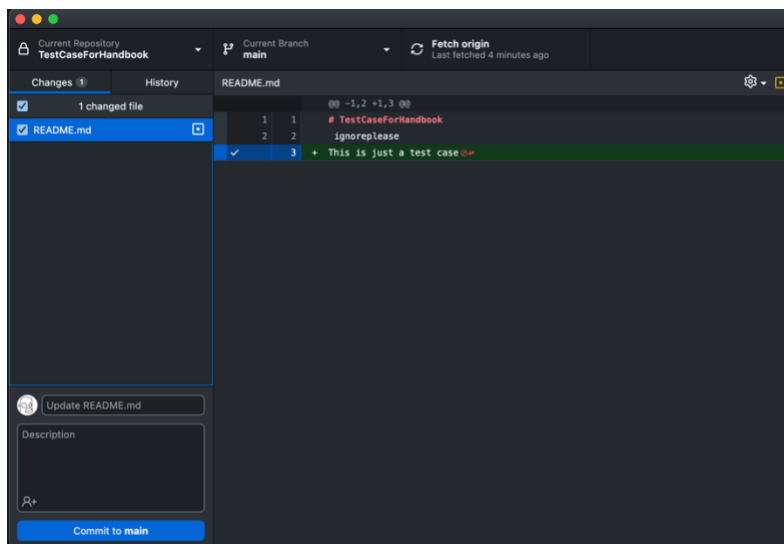
Since we will be using GitHub Desktop for this group assignment, I will teach you how to do everything within the app itself. To clone a repository first you need to have one.



In GitHub desktop click on the **Current Repository** button at the top, then click the **Add** button as seen in the picture. From there you can clone a repository, create a new repository or add an existing one. Click on the **Clone Repository** button. There a new window will pop up, there you can choose to get a repository from your selection of your repositories on github.com or add one from a URL.



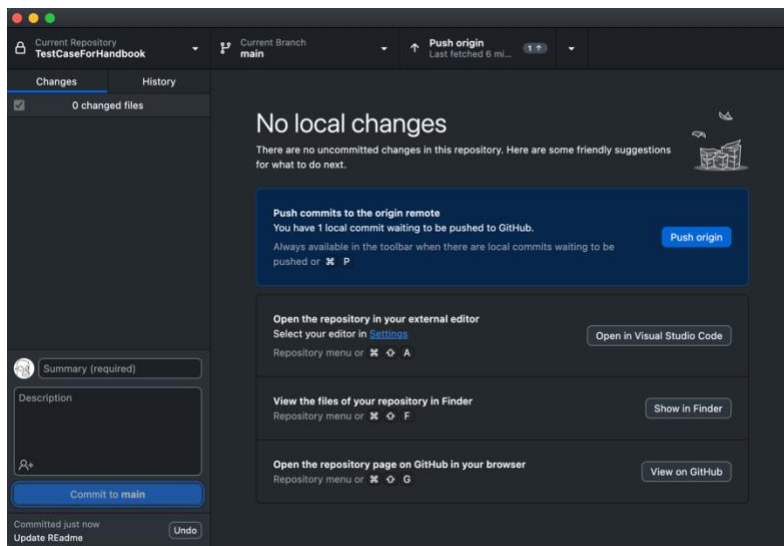
You can decide which one you want to choose, but for simplicity since I have already added you to the repository choosing from GitHub.com is simple and easy, just find the repository and click on it, then press clone and the repository is now on your computer.



After you have cloned to repo you can start making files and making your first commits. Here you can see I made changes to the README.md file. Type in a short description into the upper text box in the down left corner about what you did, for example “Updated the README” or “Added text to README” then in the description box you can go into more detail about what you did. Then just press the commit

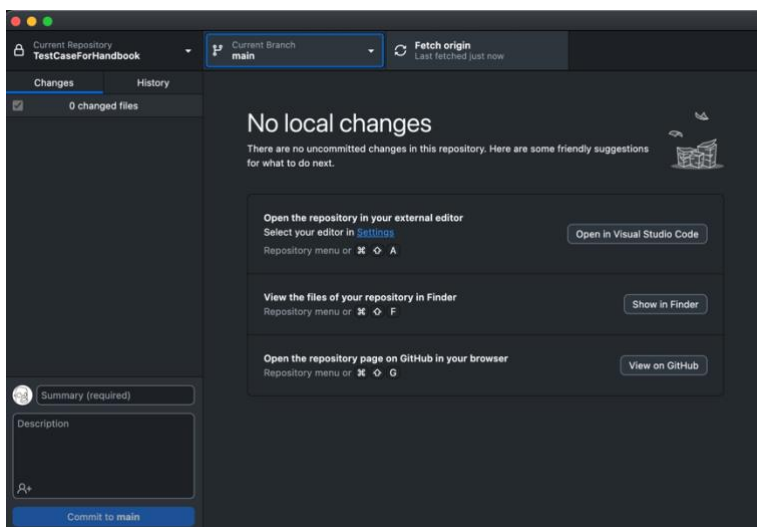
button.

## Pushing



Then you must Push the changes to the main branch or the branch you are currently on, we will go over branches later. Press the blue “Push origin” button to push the changes to the GitHub repository online, and congratulations you have pushed your first changes to GitHub.

## Pulling

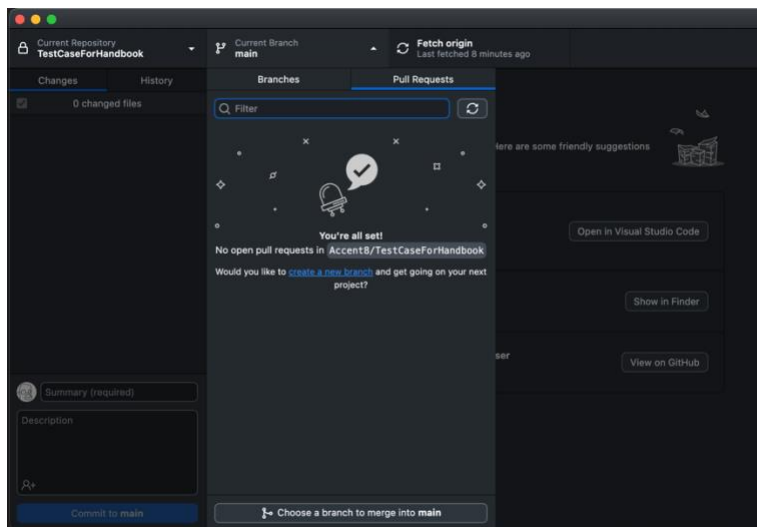


When working with a team on GitHub, members of the repository can all push their own changes to the repository maybe while you’re working on it, so I recommend at the start of everyday to “Pull” from the repository. Pull pulls all changes made to the repository, maybe ones you don’t have locally on your machine, to your machine so everyone has the same files. When someone makes

changes, I recommend you pull from the repository to get the changes to your machine. To pull from the repository just press the “Fetch origin” button at the top of your screen, and you have successfully pulled from a repository. In short you Pull to sync with the remote repository and push after making changes so other users of repository can sync with your additions.

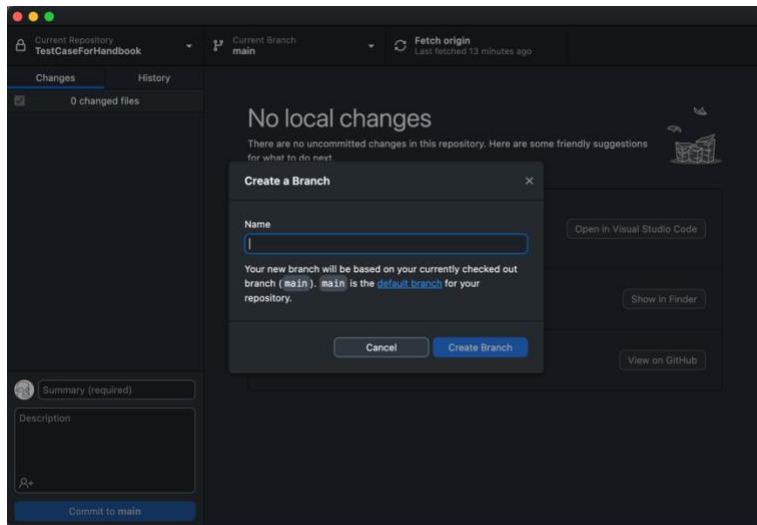
## Branching

Before we go over how to create your own branch let's start by going over what is a branch? A branch in Git is like a separate workspace where you can work on specific tasks or features without affecting the main codebase, you can think of it as a “sandbox” for making changes. Always when working on something new you want to create a branch and keep the “main” branch as clean as possible, what this means is basically while you are creating something new and it's not been tested fully keep it on a separate branch, for example: Say you are creating a login feature in our project, simply create a new branch called something like “feature-login-page” and work on the feature there. To cut it short any work you want to add to the code just put it on its own separate branch until it's complete and has been tested, there is nothing wrong with too many branches. We will be using **feature branches** where we will work on individual parts of the project like logging a user in, **development branch** where we will combine everything from the feature branches and then finally the **main branch** where we will have just complete code with no errors

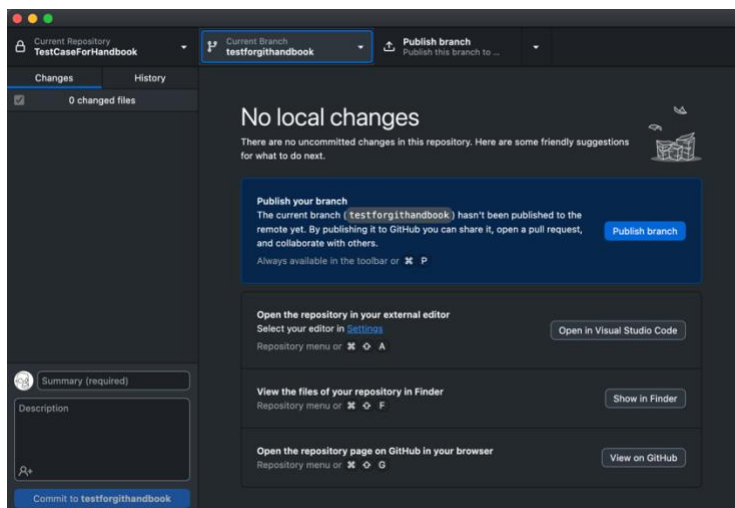


## Creating a branch

To create a branch, click on the button in the middle of the top of your screen, should be a box there that says, “Current branch” and under that it will tell you what branch you are currently on, in our case we are still on the main branch. Next once you have the menu open like in the picture click on the create new branch.



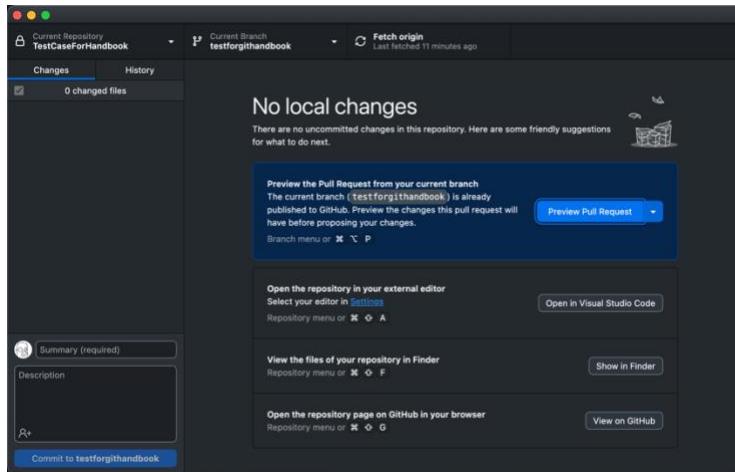
Once you click on „create new branch“ It will ask you for a name for the branch, choose a good and descriptive name for the branch like if you are working on the login feature something like „feature-login-page“ is a good name and people get a general understanding on what is going on on the branch. **Dont** name your branch something like „cool-branch-bro“



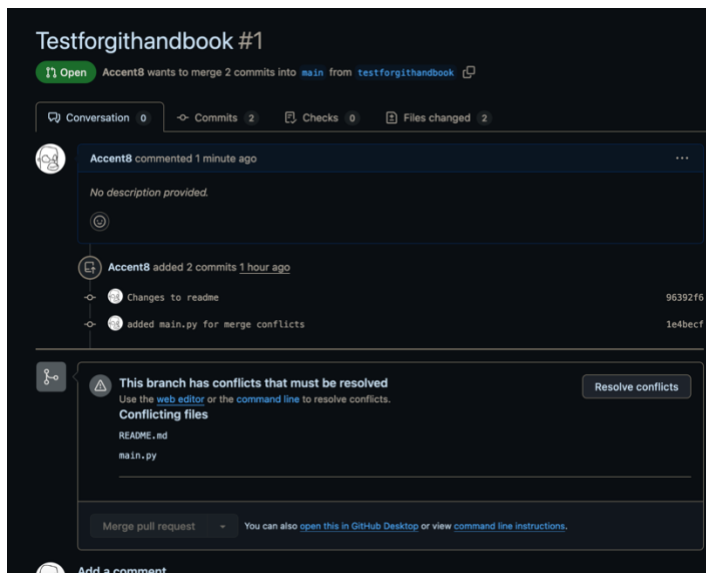
Now you need to publish the branch, what this does is similar to pushing code to the remote repository, once a branch has been pushed everyone in your team can access it. Always publish the branch right after creating it. After pressing the blue „Publish Branch“ button, and congratulations you have made your own branch. Now you can go and work on what ever you were doing.

## Merging Branches

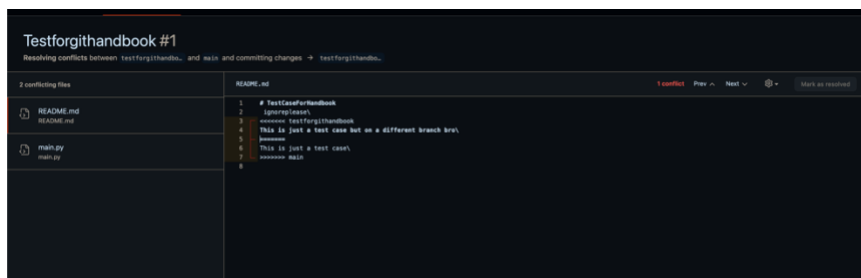
After working on your branch and adding the code you wanted to add and making sure it is all working, you have too “merge” your branch with the main branch. What that does it combines the two branches together into one. What can happen when you want to merge 2 branches is you get merge conflicts, which is a common problem and often simple to fix. When merging 2 branches we always want main to be our “base” since the code we know works is there and all the other work we have done is also there.



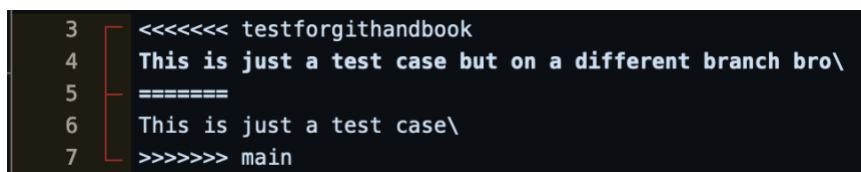
Once you have made changes on the branch, you can make a “pull request” which is basically the process of merging 2 branches together. Click on the blue “Preview Pull Request” button to merge the branches. Now either 2 possibilities have happened, either you get merge conflicts, or in the rare case you don’t you will have successfully merged your branch with the main branch. Let’s start by going over merge conflicts.



When you create a pull request and have merge conflicts you are taken over to github.com in your browser. Press the “Resolve conflicts” button.



You will get this screen. In the highlighted lines you will see where your merge conflicts are.



In the first 2 lines in this example are lines from the branch you were working on then they are split with ===== signs

Below that are the lines and what they contain but on the main branch. Now it is up to you too decide which version to keep. Keep a lookout for the conflict markers <<<<, ==, >>>>



```
README.md
1 # TestCaseForHandbook
2 ignoreplease\
3 This is just a test case but on a different branch bro\
```

Just remove lines that you dont want to keep, **and remember** to remove the split (=====  
and where github tells you which branch the lines come from. Then just press the mark as resolved

Testforgithandbook #1	
Resolving conflicts between testforgithandbo... and main and committing changes → testforgithandbo...	
Commit merge	
2 conflicting files	README.md ✓ Resolved
README.md README.md	1 # TestCaseForHandbook 2 ignoreplease\ 3 This is just a test case but on a different branch bro\
main.py main.py	

After resolving all conflicts just press the commit merge button in the top right hand corner.

### Testforgithandbook #1

Open Accent8 wants to merge 3 commits into main from testforgithandbook

Conversation 0 Commits 3 Checks 0 Files changed 2

Accent8 commented 33 minutes ago

No description provided.

Accent8 added 3 commits 1 hour ago

- Changes to readme 96392f6
- added main.py for merge conflicts 1e4becf
- Merge branch 'main' into testforgithandbook Verified d7557f5

Require approval from specific reviewers before merging

Continuous integration has not been set up

This branch has no conflicts with the base branch

Merge pull request

### Testforgithandbook #1

Open Accent8 wants to merge 3 commits into main from testforgithandbook

Conversation 0 Commits 3 Checks 0 Files changed 2

Accent8 commented 34 minutes ago

No description provided.

Accent8 added 3 commits 1 hour ago

- Changes to readme 96392f6
- added main.py for merge conflicts 1e4becf
- Merge branch 'main' into testforgithandbook Verified d7557f5

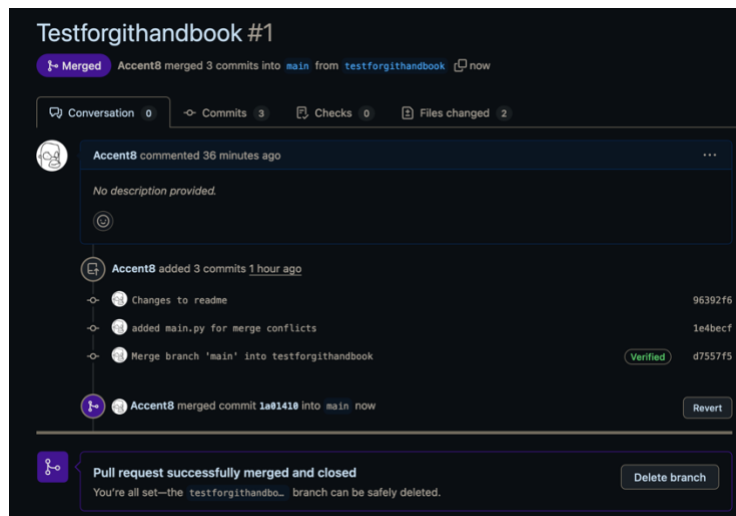
Merge pull request #1 from Accent8/testforgithandbook

This commit will be authored by 70168436+Accent8@users.noreply.github.com

Confirm merge Cancel

When you have solved all conflicts, and if you had no conflicts you will get to this page. Press the „Merge Pull Request“ button

Then press the confirm merge button.

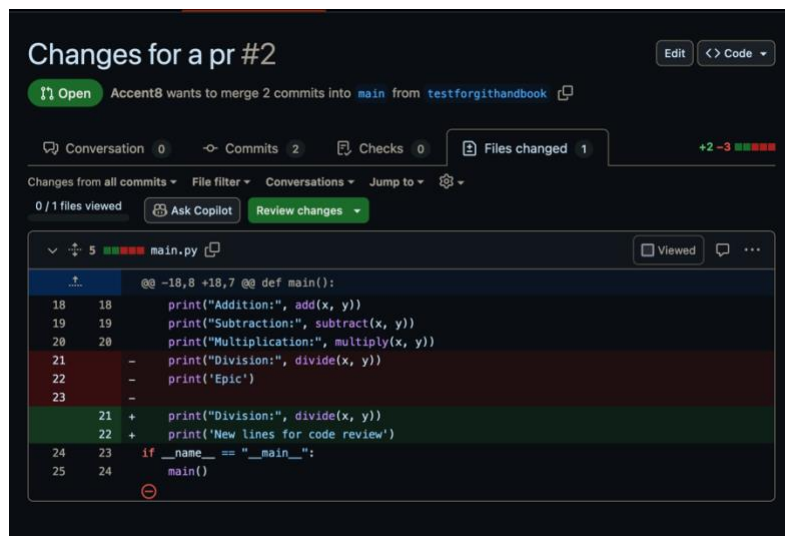


And then you have successfully merged 2 branches. Now you can delete the branch if you want to (Recommended)

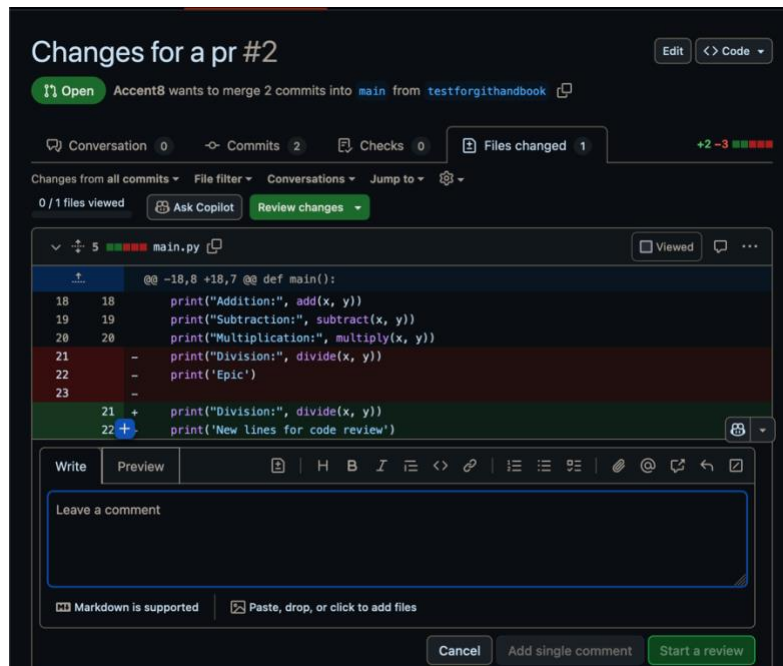
## Code Reviews

Code review is a process where team members examine code changes made by a developer to ensure it meets quality standards, follows best practices and aligns with the project's requirements. All pull requests and merges we will do will be required to have a code review. Why are they important? They catch errors early, improve code quality, helps team members understand different parts of the codebase, maintains consistency and we can learn from each other's feedback. We do code reviews after a pull request is submitted and before merging any changes to the **Development** and **Main** branches.

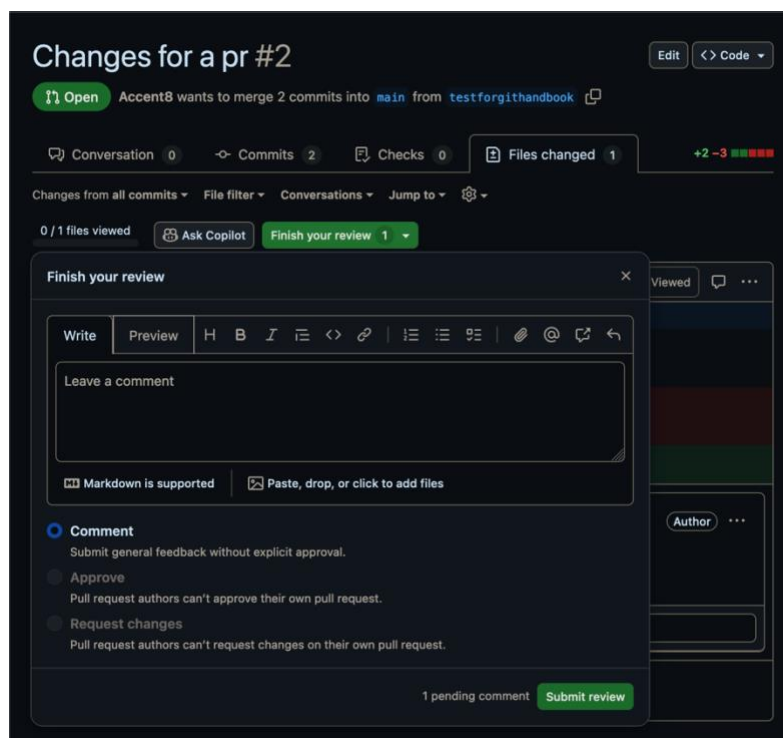
How to perform a code review in GitHub.



Open a pull request and head to the files changed tab like in the picture, now you can go over any line and there should be a plus icon that appears next to the line number. Click on it



Type in your comment for example “Consider renaming this variable to make its purpose clearer.” Once you are satisfied with your comments. Click on start a review



Once you have finished your review, click on the finish review button. You can decide now what you want to do, you type in a comment then you just comment it, just like general feedback. You can approve the pull request, then the pull request would go through, though you **cannot** approve your own pull request, or you can Request changes, where the author will get the feedback and must make changes before making another pull request. Once you have decided what needs to be done just press the submit review.

## Branch Strategies

We will be using 2 main branches; those will be **Main** and **Development**. On the main branch we want stable, production-ready code. Development branch is for integration new features, making sure they work together before merging into main. Then we have **feature**

branches, these branches you will create for individual tasks, and you will branch these out from the development branch

## Common Problems and Solutions

You will most likely run into some problems over the course of these 3 weeks. Some of these problems could be

1. Unable to push changes.
  - a. A solution to this is to ensure you have the latest changes from the remote repository by pulling first. At the start of everyday I recommend you start by pulling from the remote repository
2. Authentication issues.
  - a. Solution is to just re-login, do this by going to File > Options in the GitHub desktop app
3. Merge Conflicts
  - a. Solution, go over everything in the merge conflict section of this handbook
4. Accidentally deleting a branch
  - a. If the branch existed on the remote repository, you could restore it on github.com under **Branches > Closed Branches > Restore**
5. Overwriting Changes
  - a. Avoid force pushes unless necessary.
6. Too many Active Branches
  - a. Just regularly delete stale branches after merging
7. Poor Commit Messages
  - a. Write messages that explain **what** and **why** not just how.