Introduction to GitHub

Next Slide

<h1> Zakhar Dziublyk and Nikita Bratus </h1>

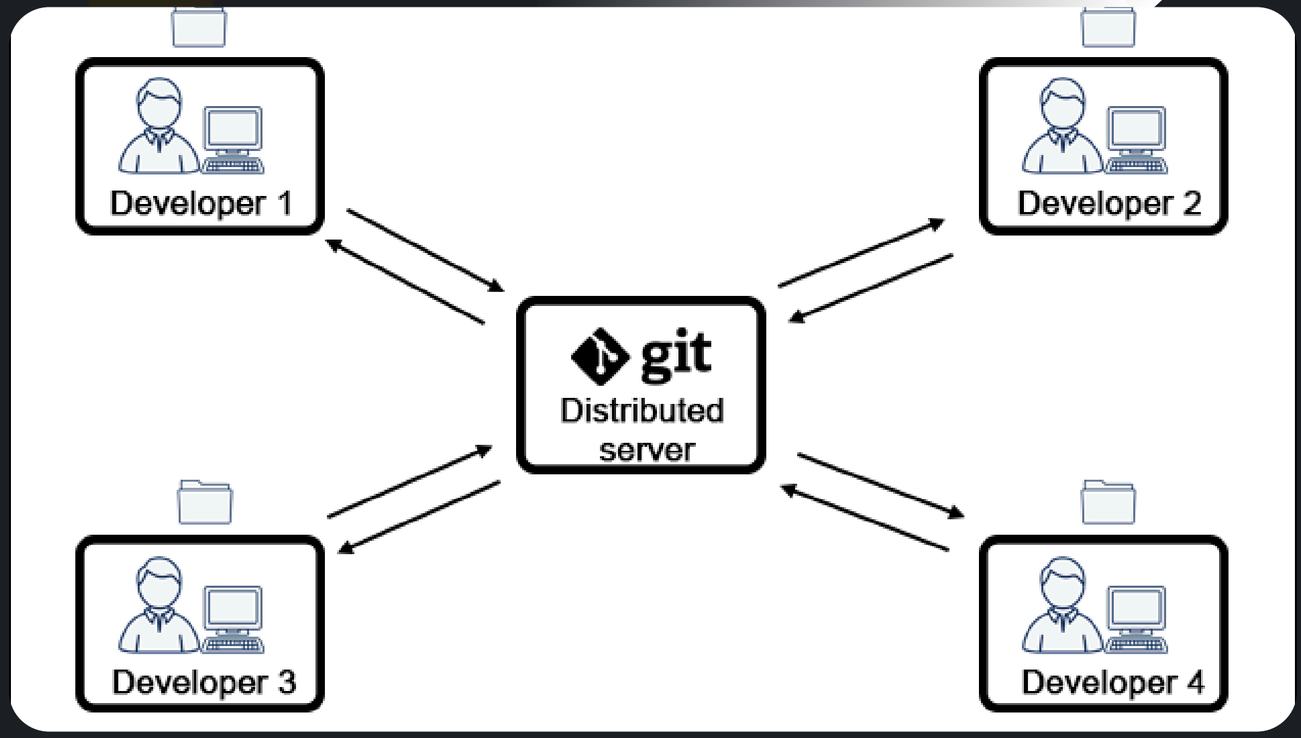
-

TODAY’S MENU

– What is Git?
– What is GitHub?
– Local vs Remote and basic Git operations
– Visual studio & GitHub Desktop
– Practice basic operations
– .gitignore file
– What are branches?
– Practice creating branches
– What is a pull request?
– Practice creating pull request
– Reviewer responsibilities
– What is merge and how to solve merge conflicts?
– Practice on resolving merge conflicts
– Writing a good README.md
– Using GitHub issues and Projects
– Closure

WHAT IS GIT?

Git is a version control system that records every change you make to your project.
It allows you to roll back mistakes, experiment freely, and see exactly how your code evolved over time.
Think of it as a timeline of reliable save points.



WHAT IS GITHUB?

GitHub is the online home for your Git repositories.
It allows you to store your projects safely in the cloud, collaborate with teammates, review each other's work, and build a portfolio that future employers can actually see.

- Access your code from any device
- Work with teammates
- Review each other's changes
- Share projects with teachers or employers

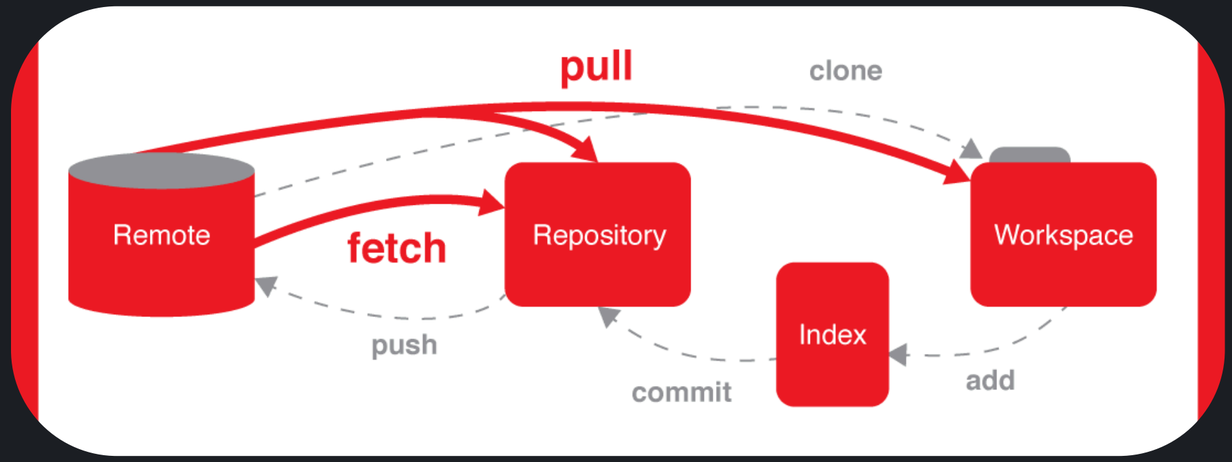
The founders of GitHub are Tom Preston-Werner,Chris Wanstrat and.P. J. Hyett

Simon Oxley is the creator of the famous GitHub logo

LOCAL VS REMOTE

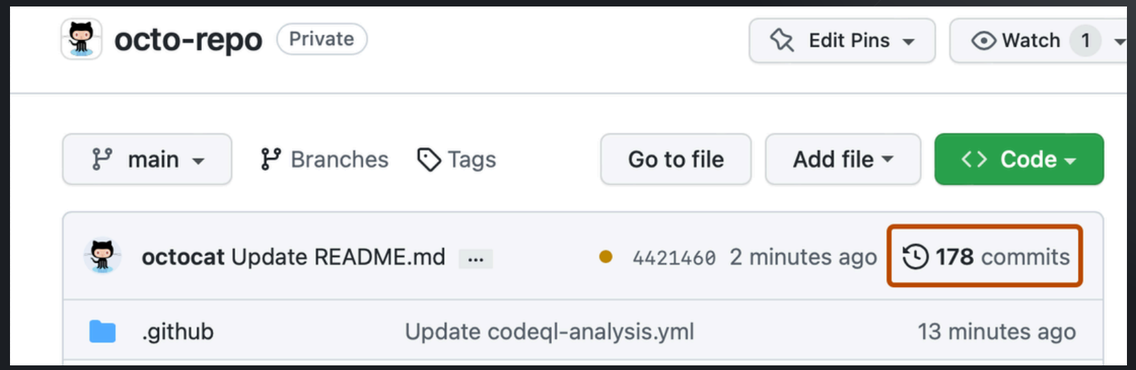
Every Git project has two versions: the local copy on your computer and the remote copy stored on GitHub.
You connect these two versions by committing your changes locally, pulling new work from others, and pushing your updates to GitHub so the team stays in sync.

Local Repository: Your personal copy on your laptop
Remote Repository: The shared version stored on GitHub



Commit → save changes locally
Pull → download updates from GitHub
Push → upload your work
Fetch → check for updates without applying them

UNDERSTANDING COMMITS

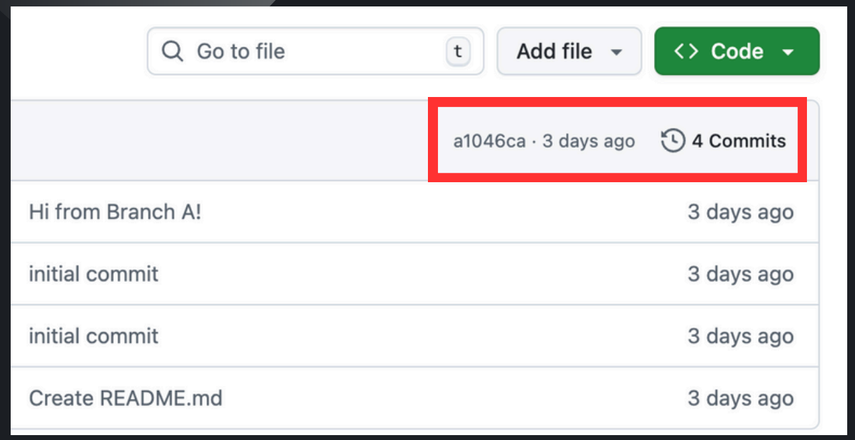


Creating a Save Point in Your Project

A commit is a saved snapshot of your project at a specific moment in time.

Think of it as: Taking a photo of your project

INDEXING IN SIMPLE WORDS



Staging is the step where you choose which changes will be included in the next commit.

Preparing Changes for Commit
- Happens before commit
- You don't have to commit all changes
- Gives you full control

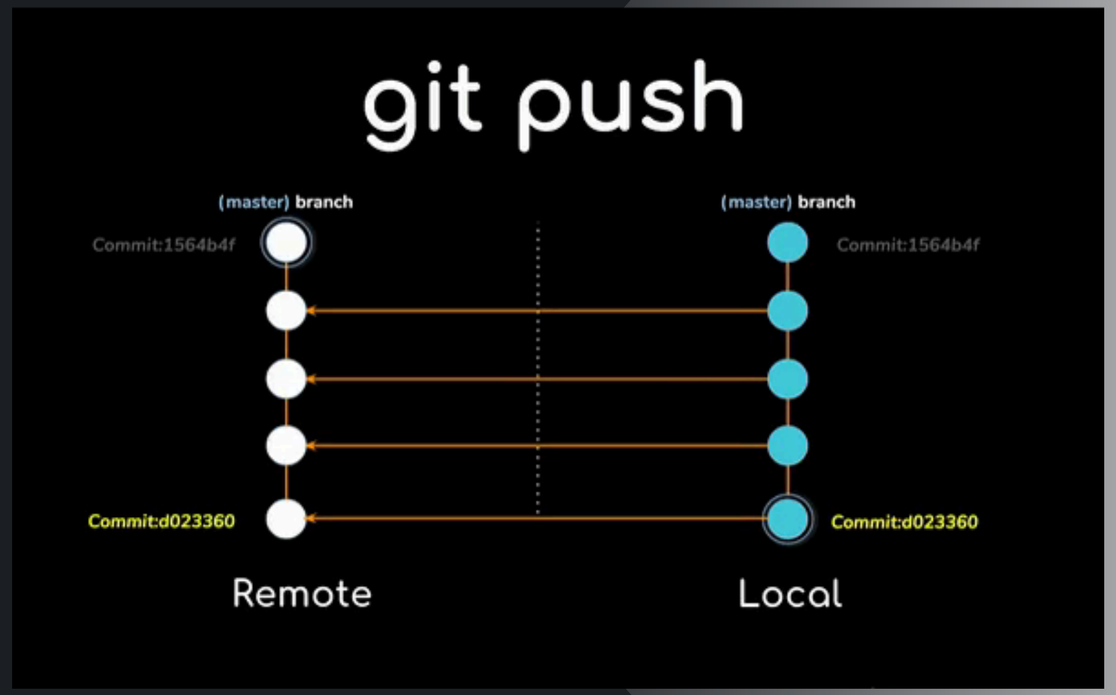
Flow:
Working Directory → Staging Area → Commit

PUSH PROCESS

Sharing Your Work with the Team

Push sends your local commits to the remote GitHub repository.

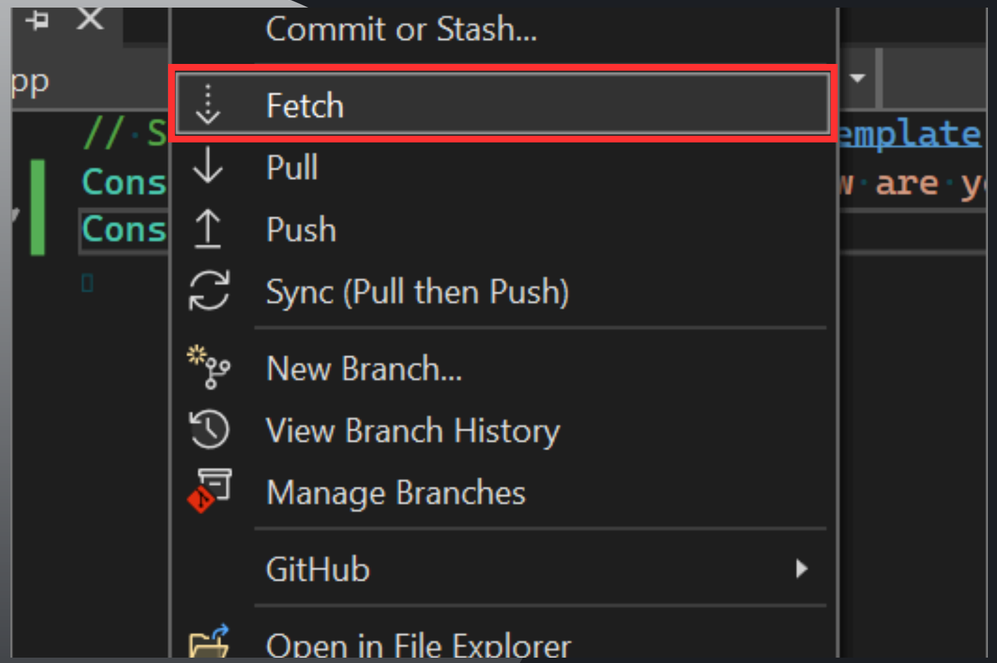
- Uploads changes to GitHub
- Makes them visible to others
- Might be blocked on protected branches



Flow:
Your Computer → GitHub

WHAT IS FETCH AND HOW TO USE IT

Checking for New Updates



Checking for New Updates
Fetch downloads information about new commits from GitHub without changing your files.

– Only updates your history
– Safe operation (no local code changes)
– Helps you stay informed

Think of it as: “Let me see what changed”

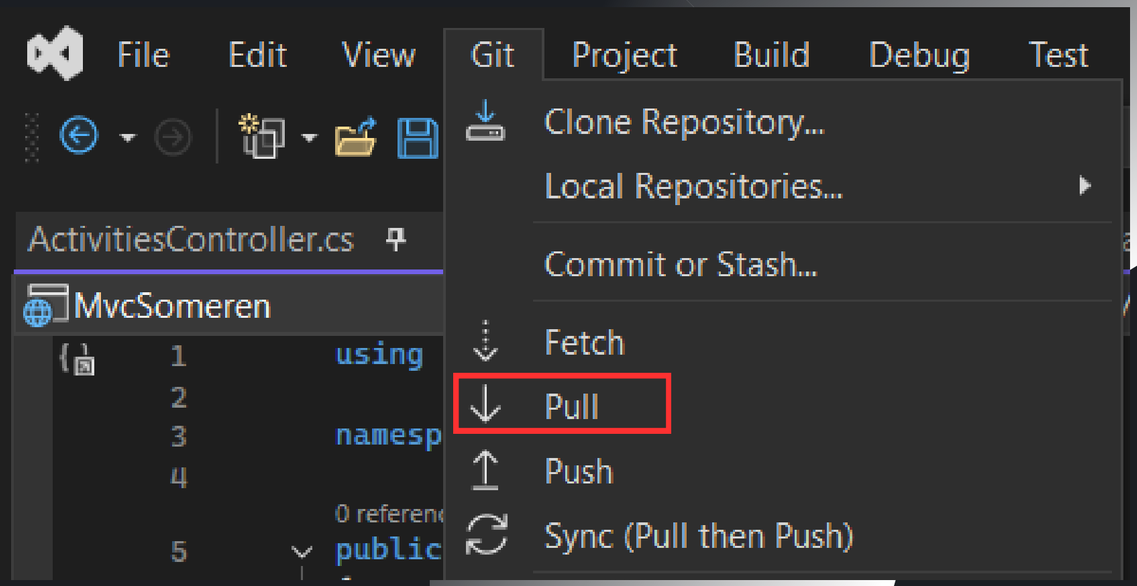
WHAT IS A PULL

Updating Your Local Project

Pull gets the newest changes from GitHub and applies them to your local code.

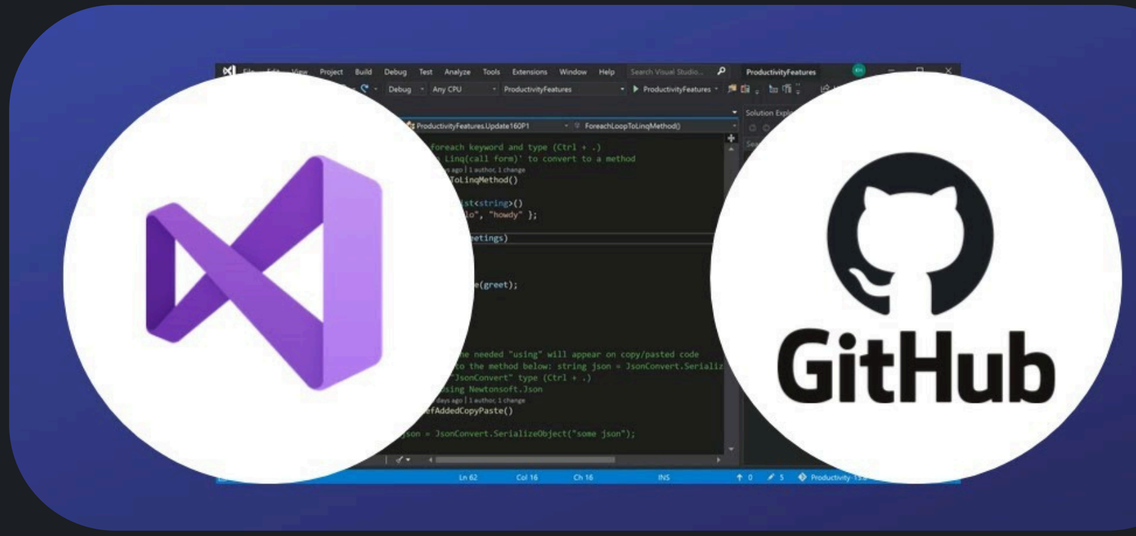
– Fetch + Merge
– May cause conflicts
– Keeps your project up to date

Think of it as: “Give me the latest version and insert it”



VISUAL STUDIO + GIT

Visual Studio includes built-in Git tools that automatically detect changed files, show you what needs to be committed, and allow you to push, pull, and manage branches without touching the command line.
This makes Git far easier and more visual, especially for beginners.

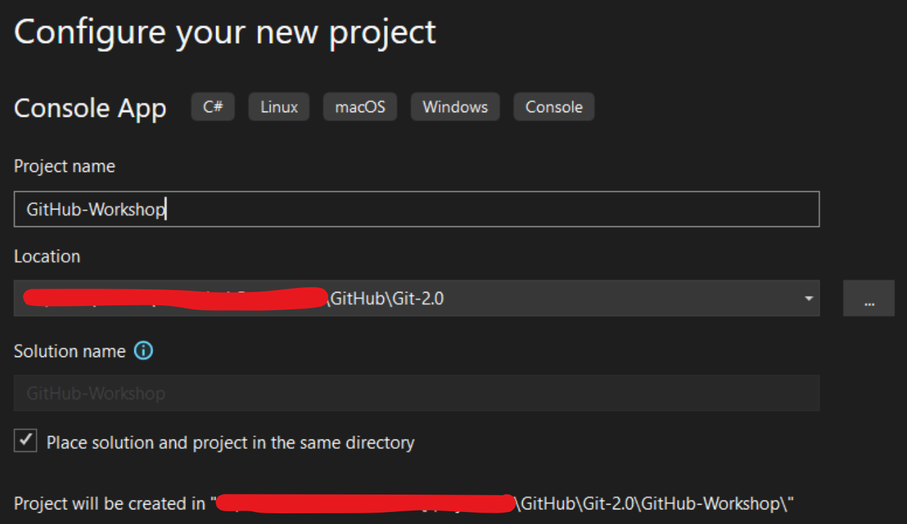


Visual Studio provides built-in Git tools:

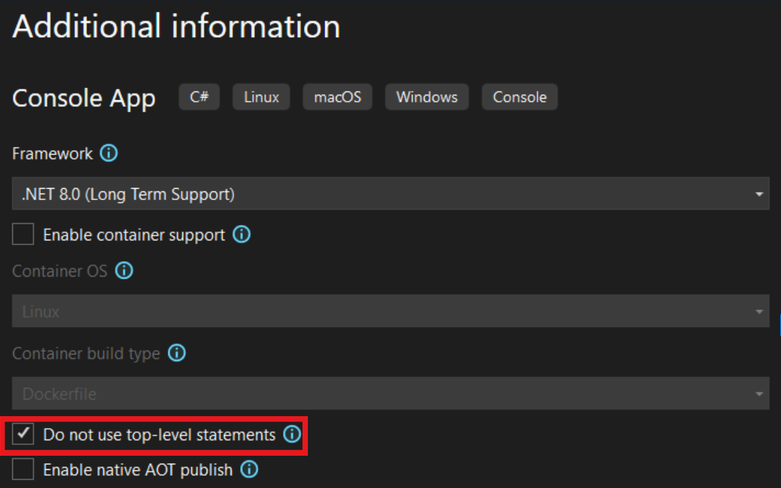
- “Git Changes” window detects modified files
- Commit, Pull, Push buttons in the toolbar
- Branch creation directly in the UI
- No command line required

SETTING UP A GIT REPOSITORY

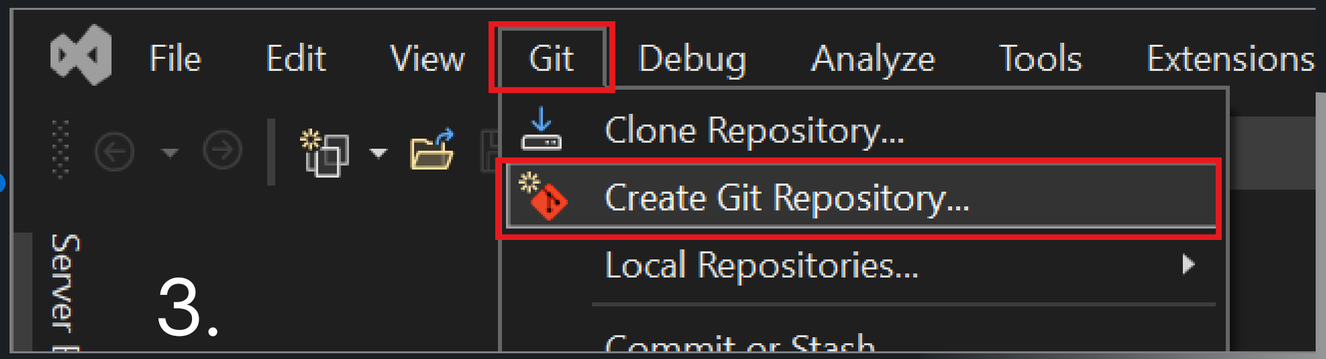
1.



2.

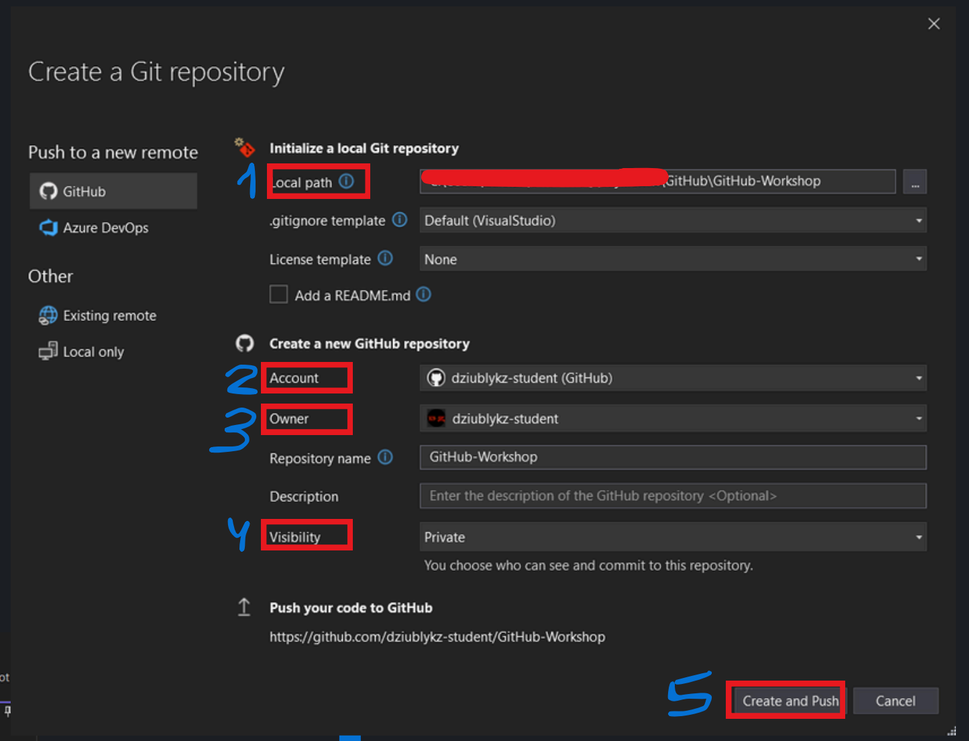


3.

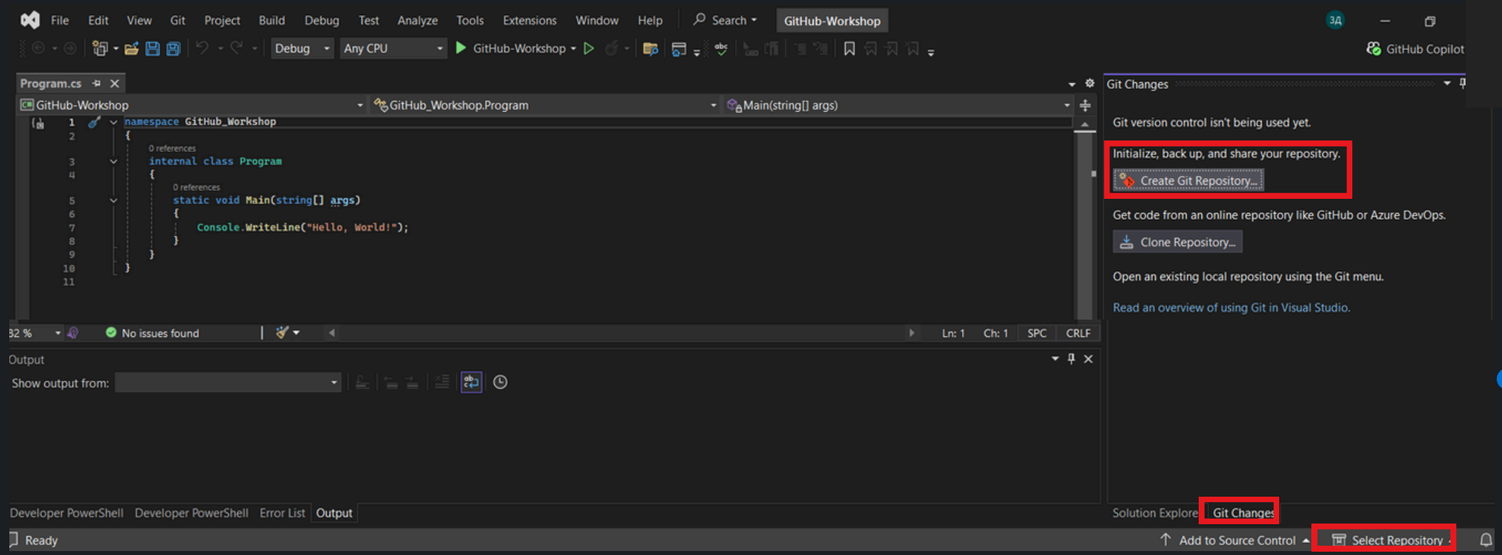


SETTING UP A GIT REPOSITORY

4.

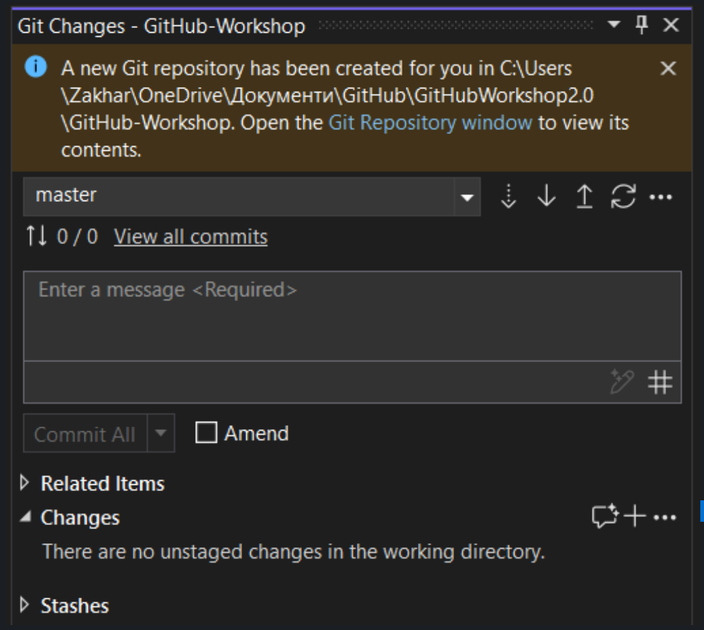


5.

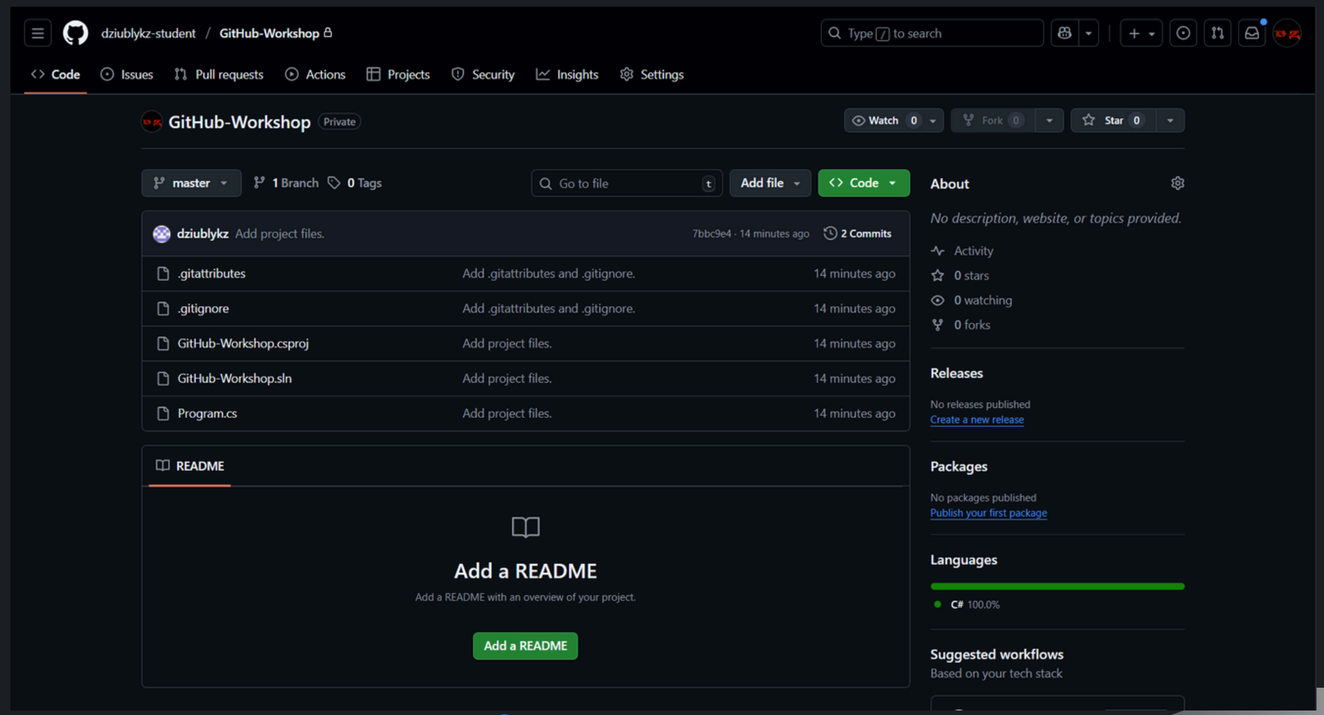


SETTING UP A GIT REPOSITORY

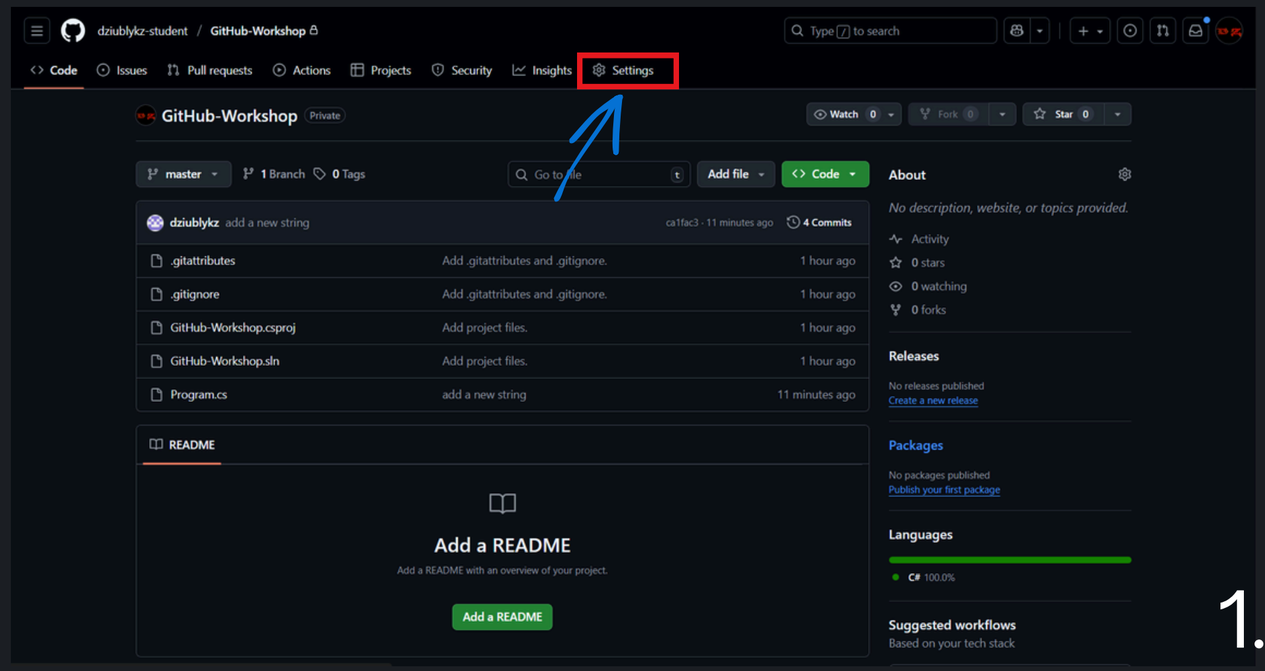
6.



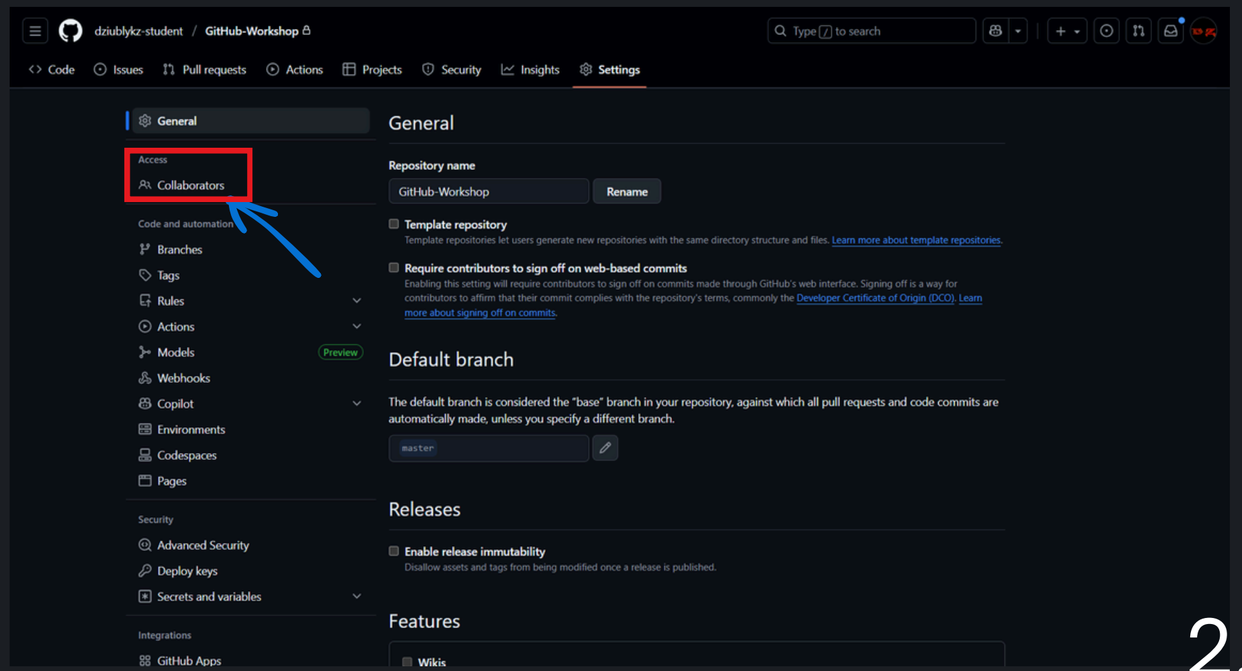
7.



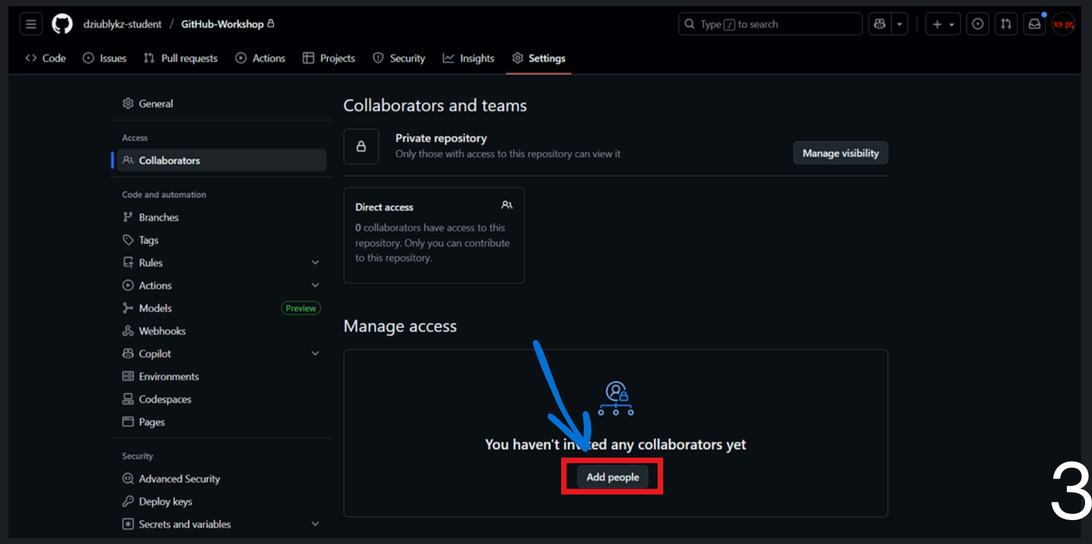
COLLABORATORS



1.



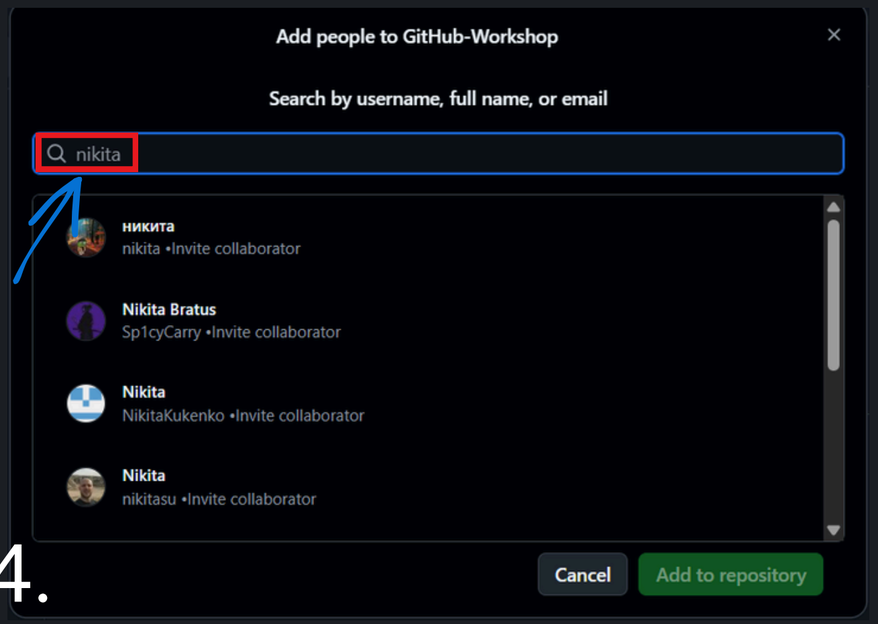
2.



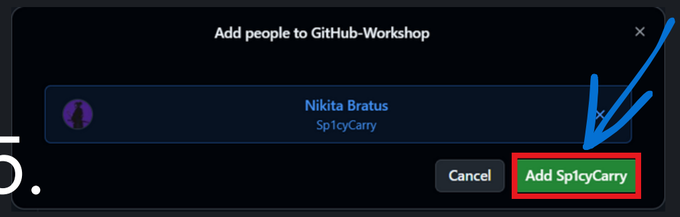
3.

COLLABORATORS

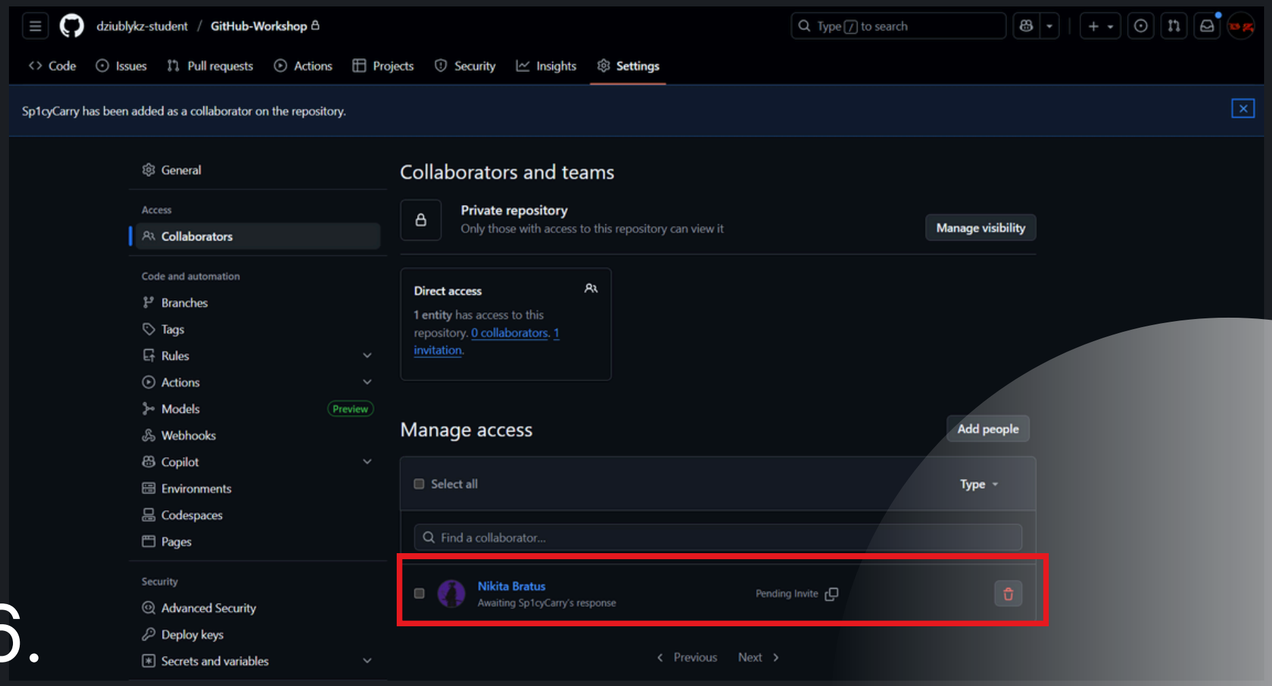
4.



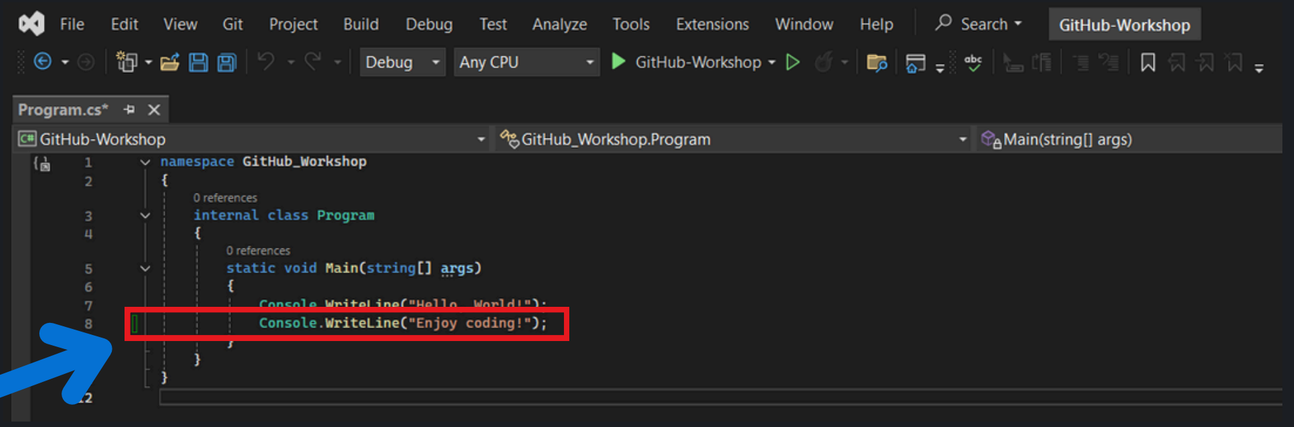
5.

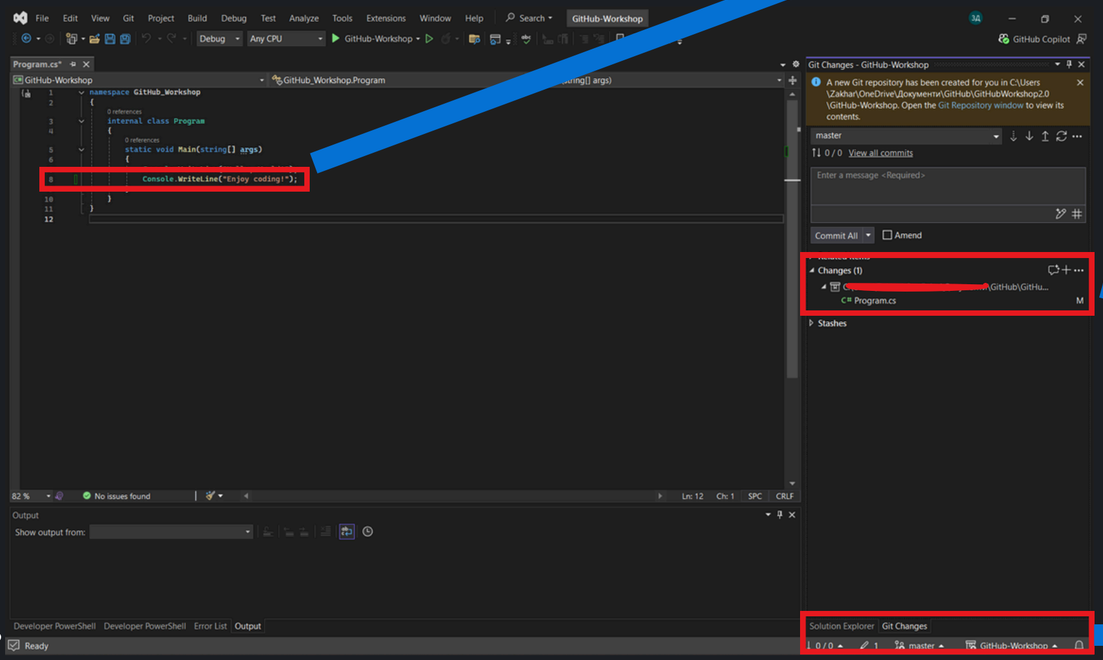


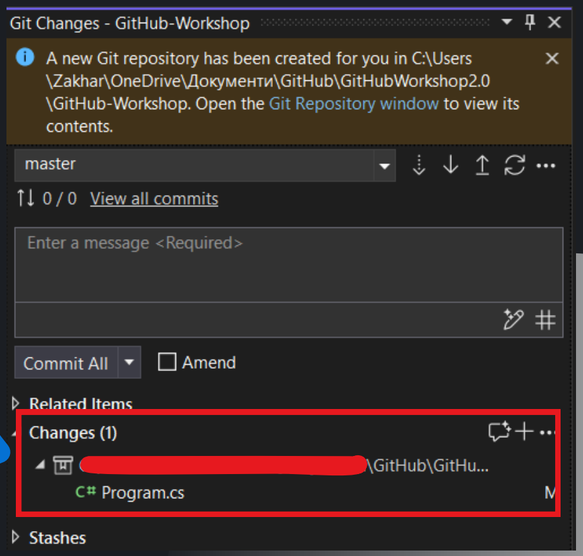
6.

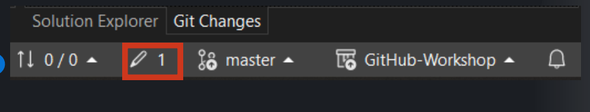


COMMITTING AND PUSHING



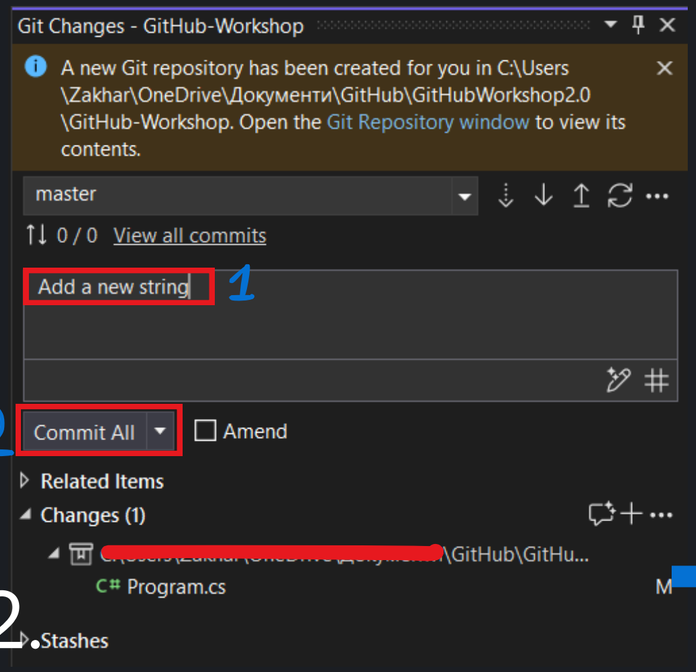


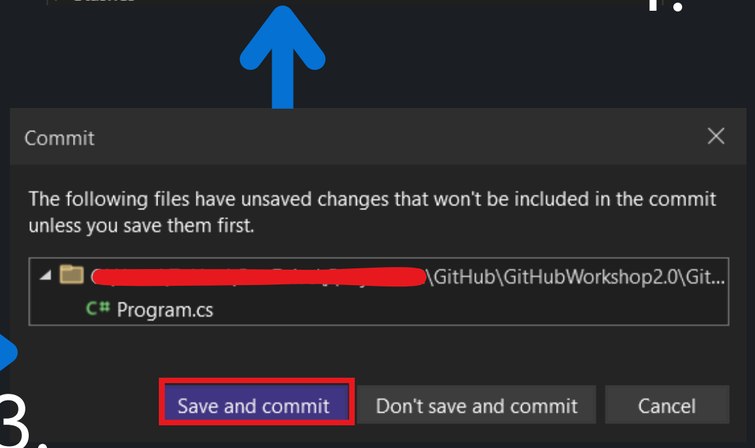


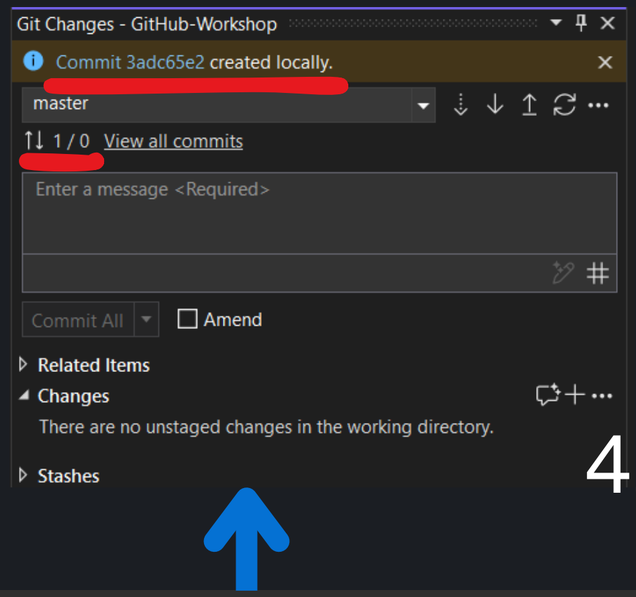


1.

COMMITTING AND PUSHING

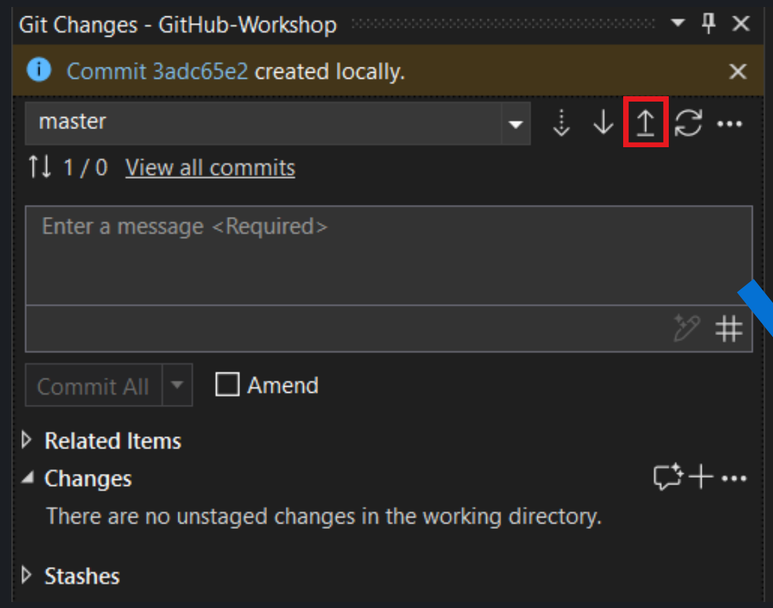


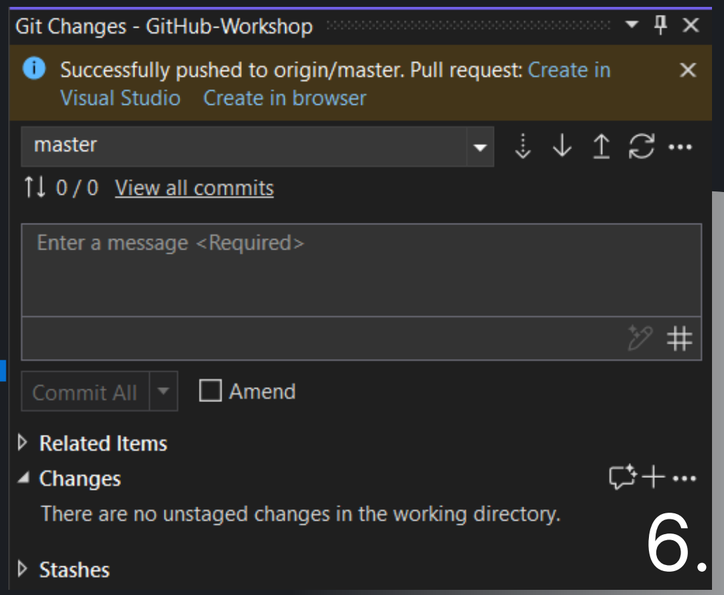




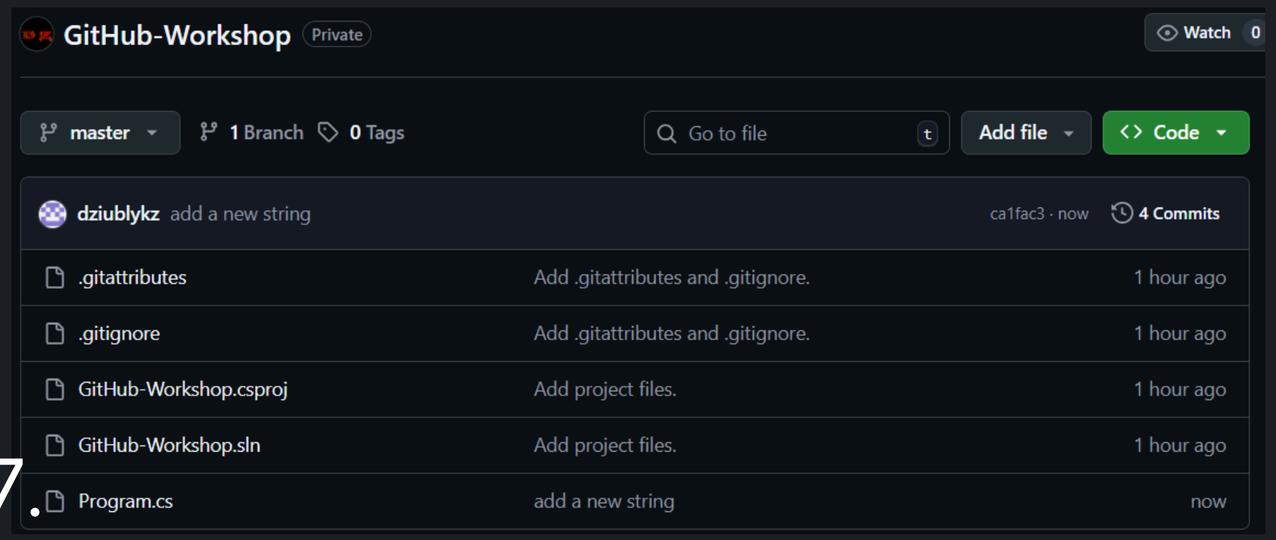
COMMITTING AND PUSHING

5.



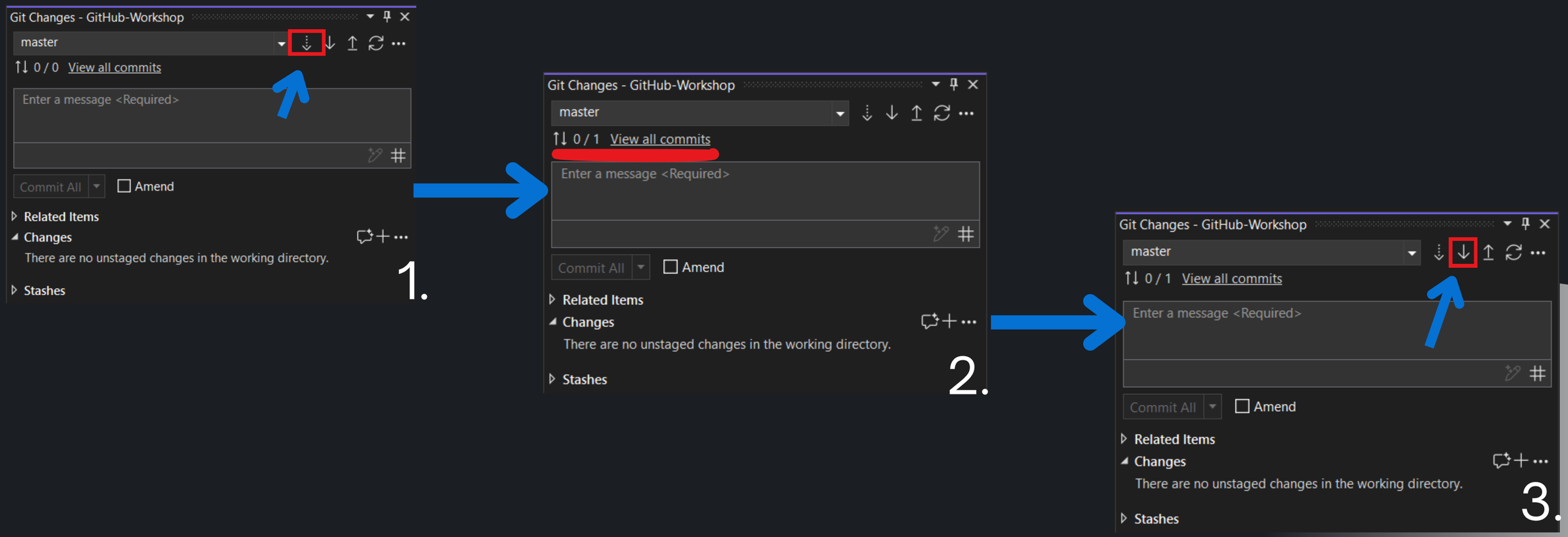


6.

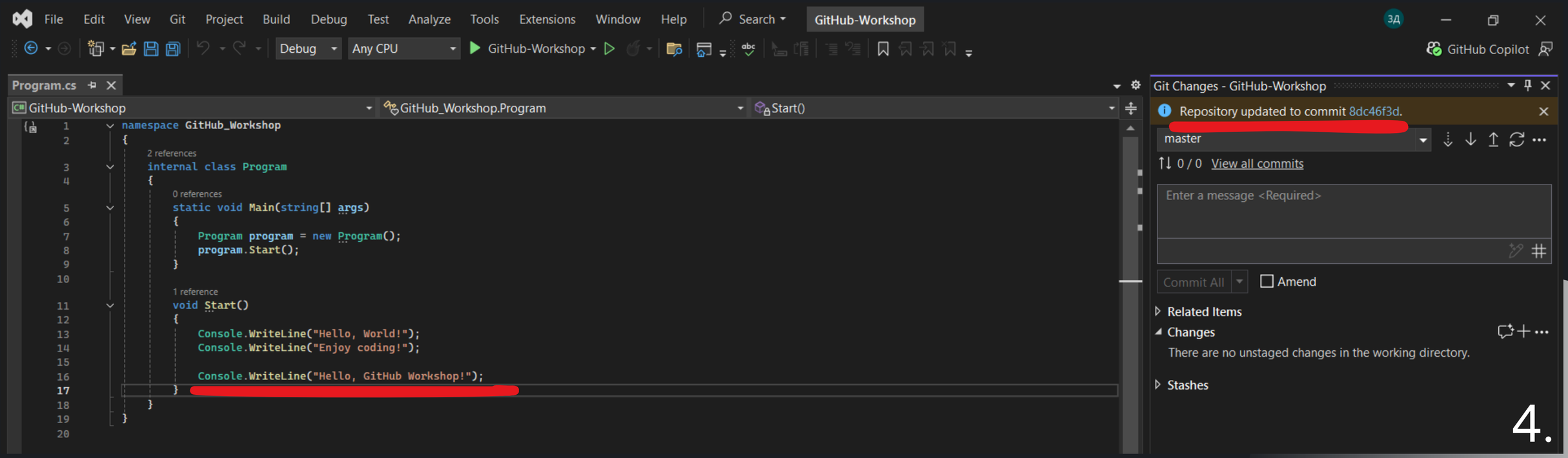


7.

PULLING CHANGES



PULLING CHANGES

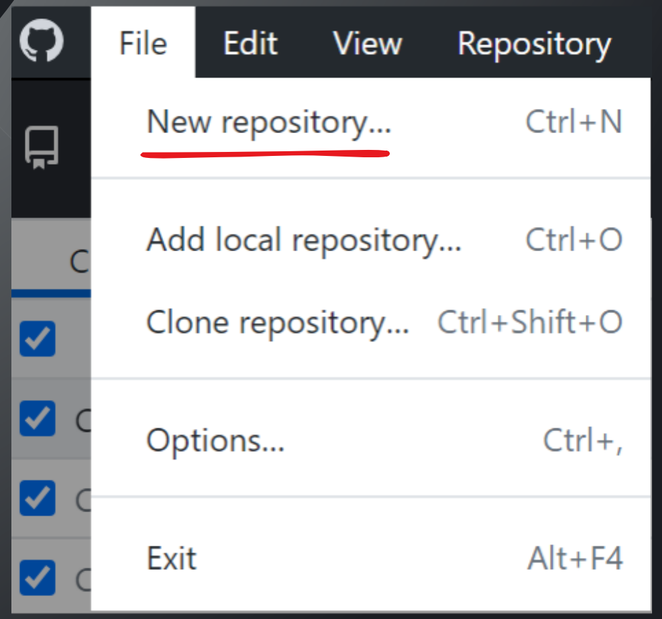


GITHUB DESKTOP

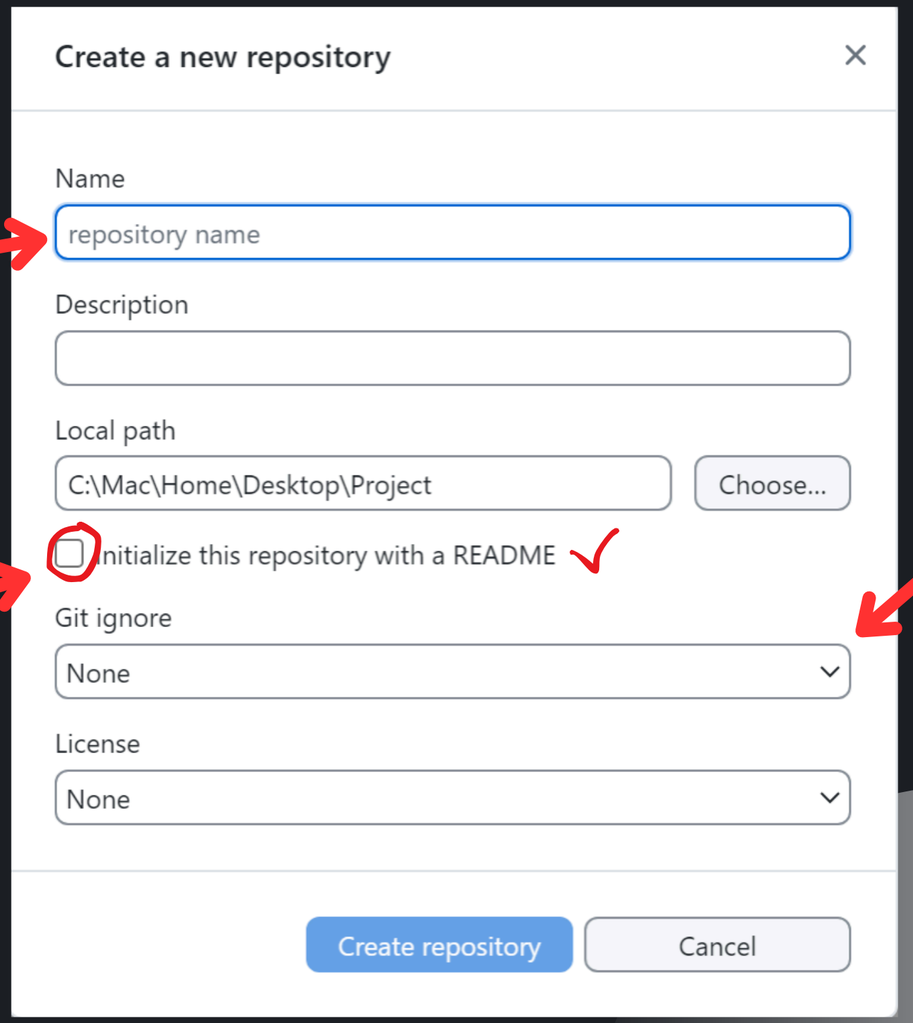
– Open GitHub Desktop, click “File → New Repository.”
– Add a name, description, and create a .gitignore or license file.
– Repository appears locally.

CREATING A REPOSITORY IN GITHUB OPTION 1

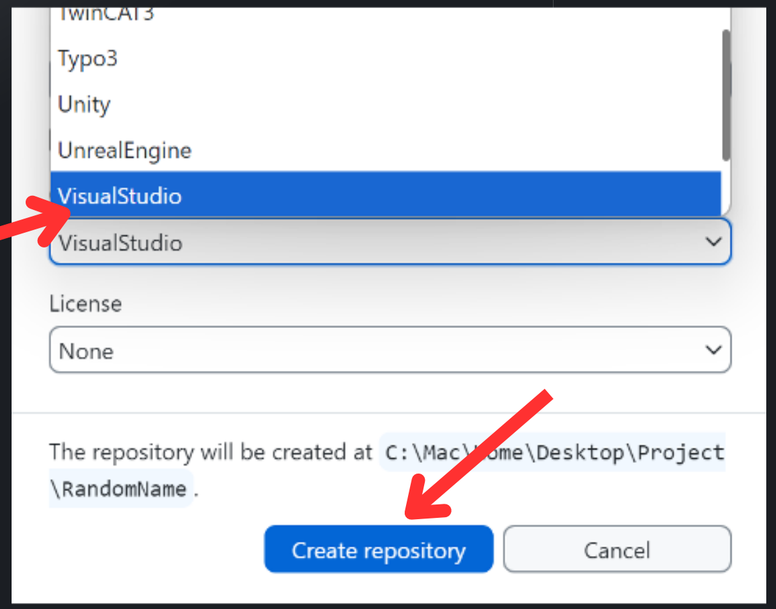
1.



2.



3.



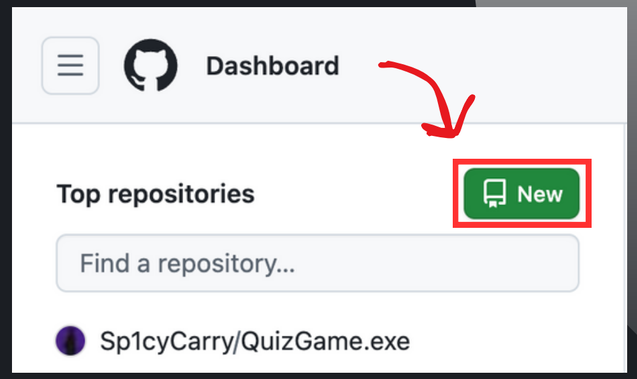
.GITIGNORE

.gitignore tells Git which files and folders should not be tracked or uploaded to GitHub.

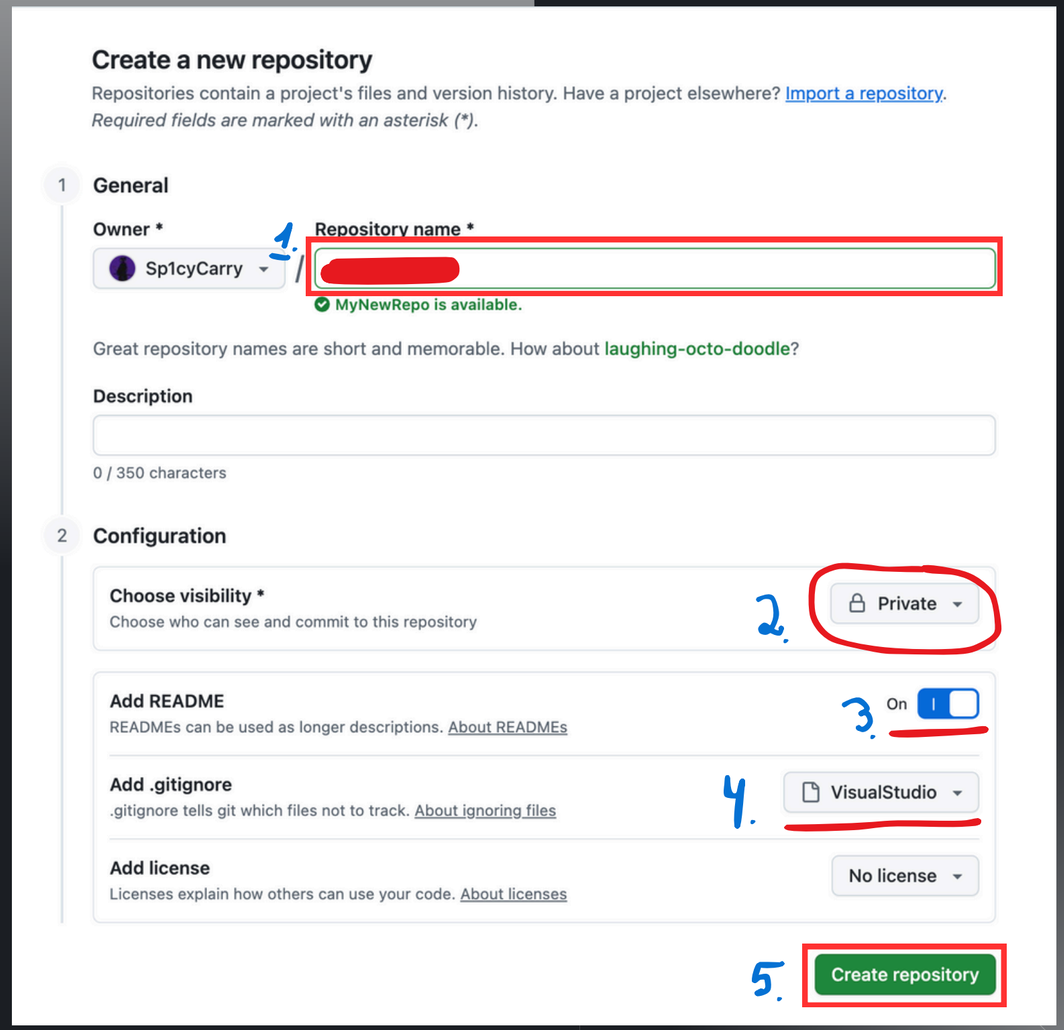
– Used for build files, logs, secrets, temp files
– Keeps the repository clean
– Can be edited at any time

CREATING A REPOSITORY IN GITHUB OPTION 2

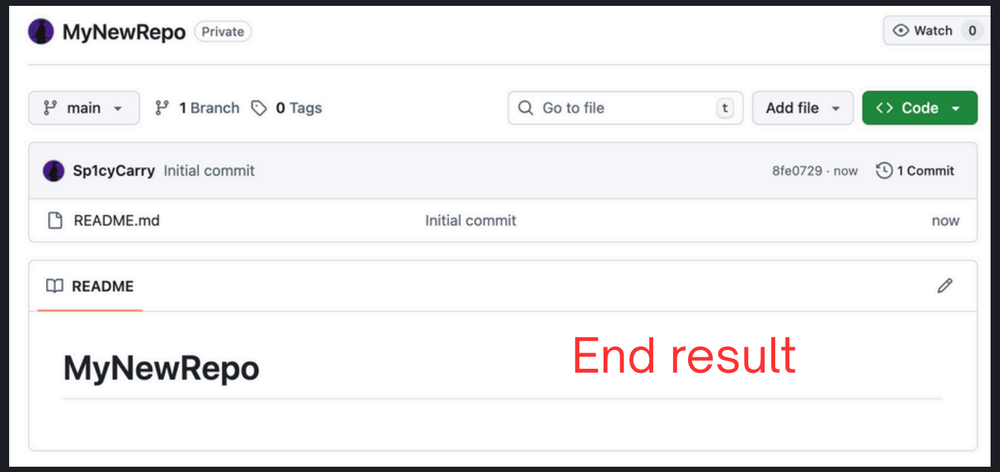
1.



2.

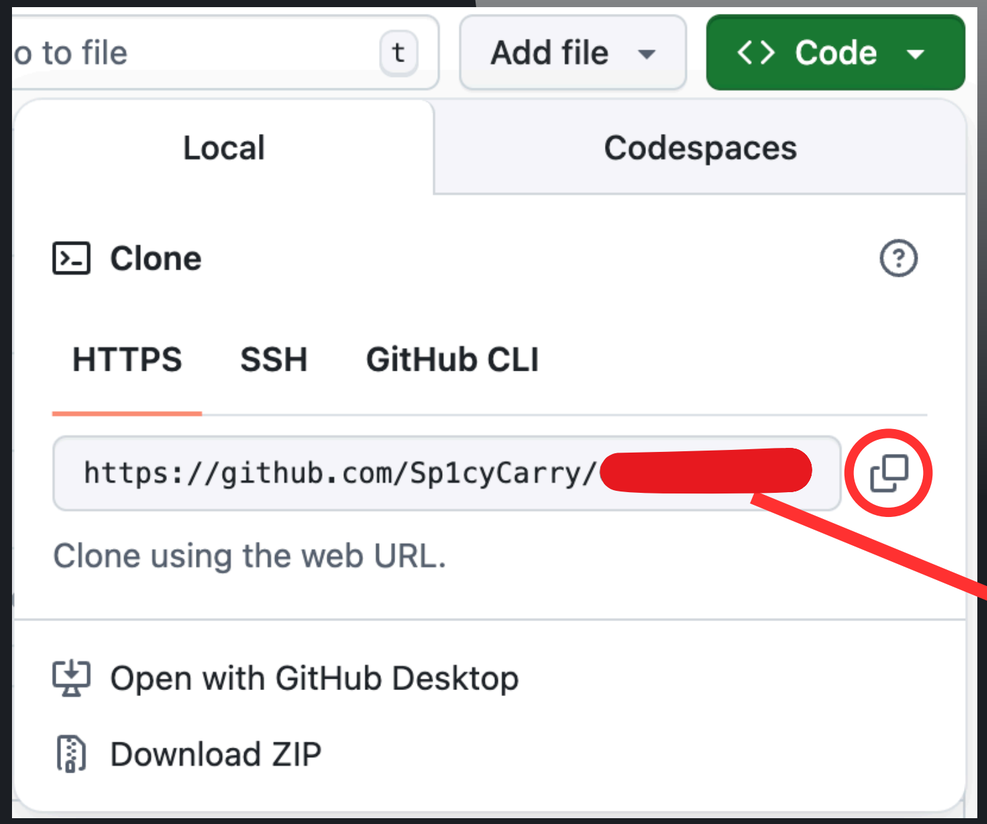


3.

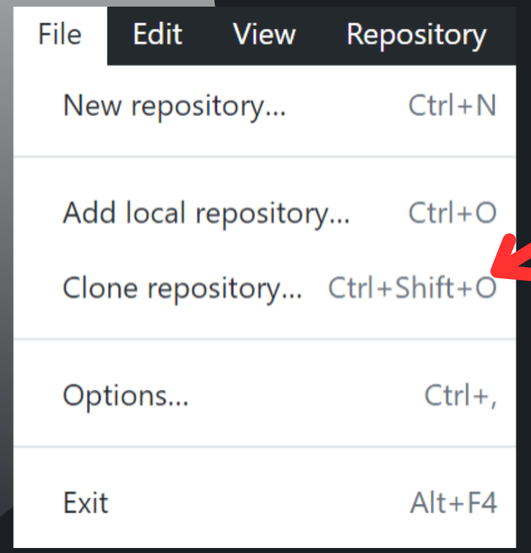


CREATING A REPOSITORY IN GITHUB OPTION 2

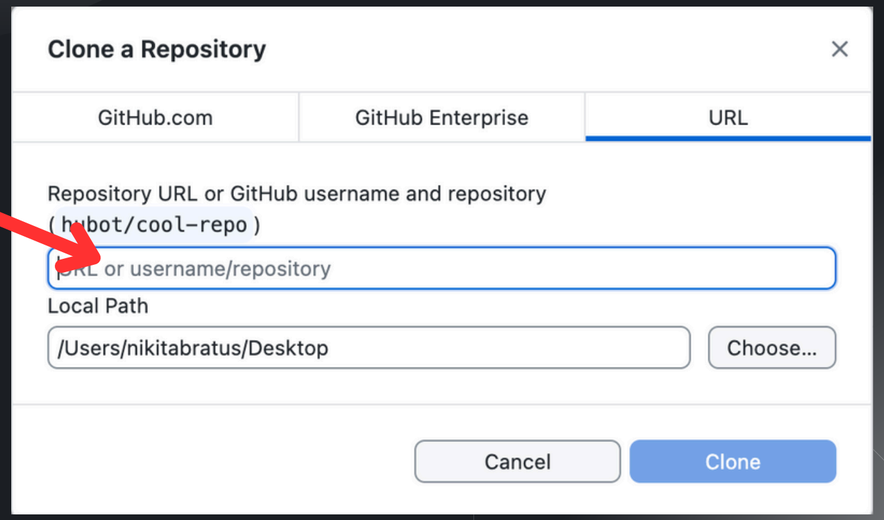
1.



2.

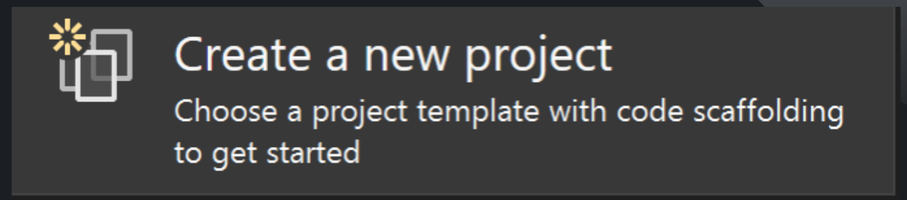


3.



CONNECTING EMPTY REPO TO THE SOLUTION

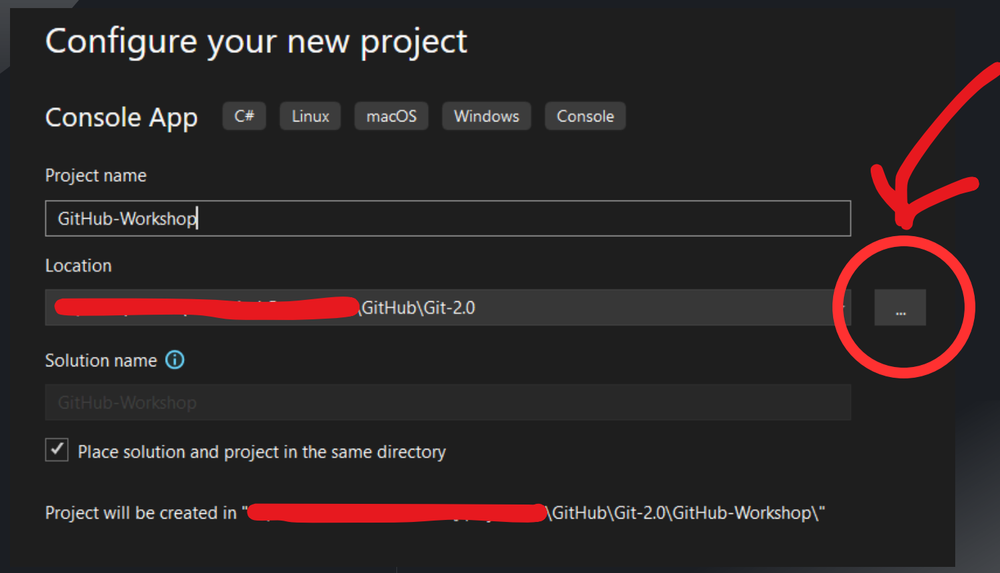
Adding files where we can change something:



Make just a small change:

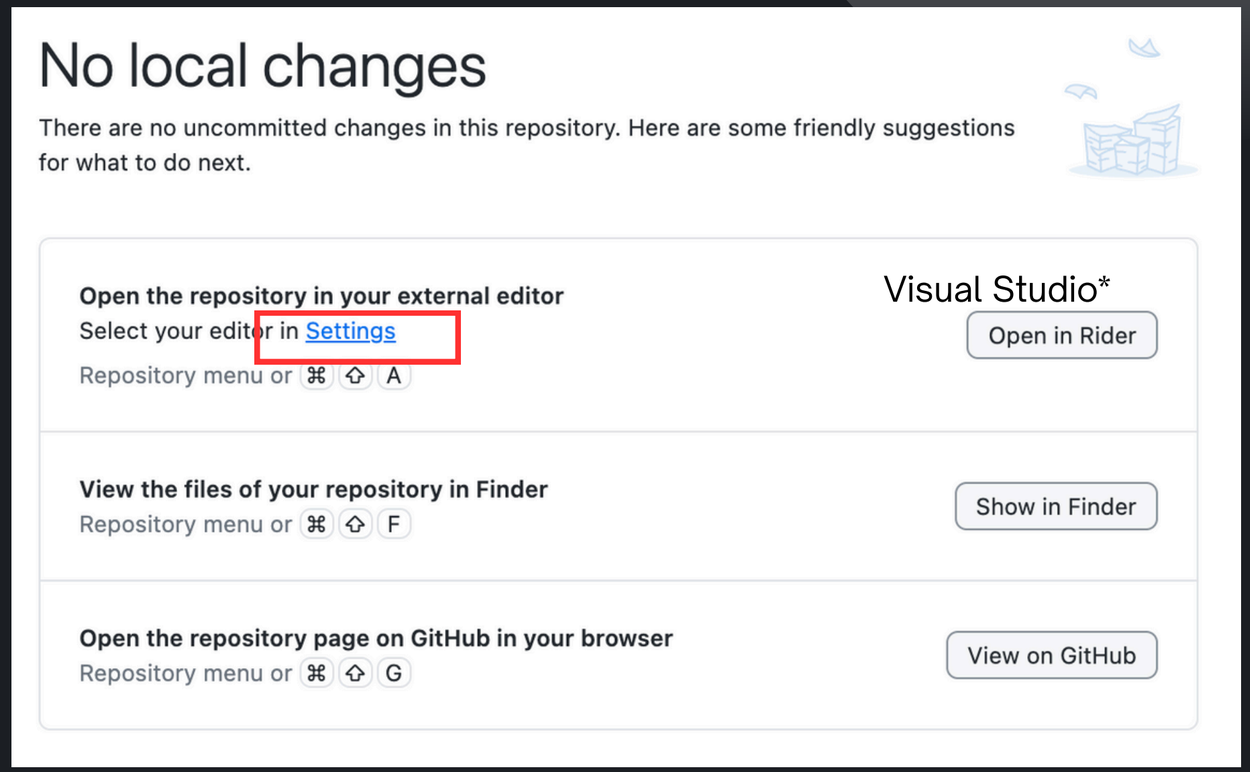
```
Console.WriteLine("Hello World!");

Console.WriteLine("I am hungry!!!");
```

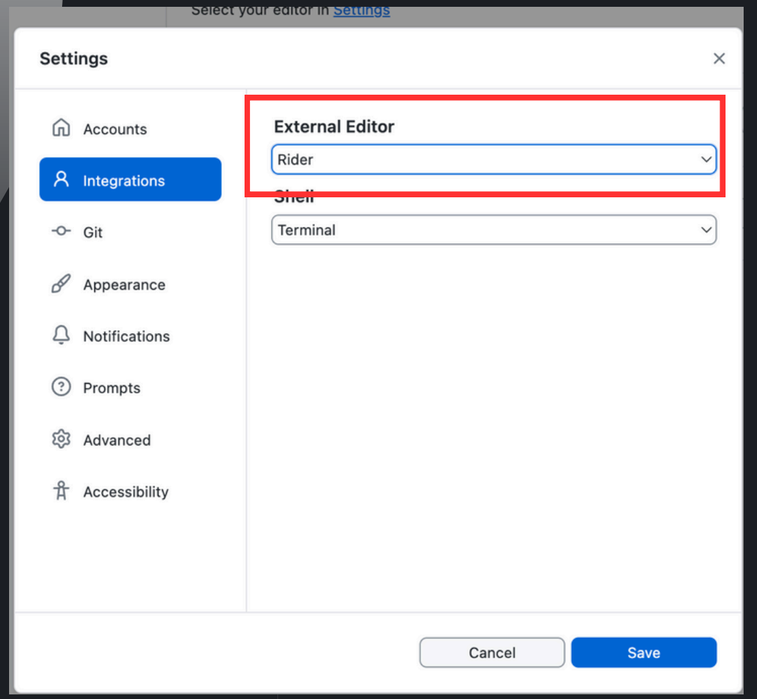
Choose the directory of the repository that you created:

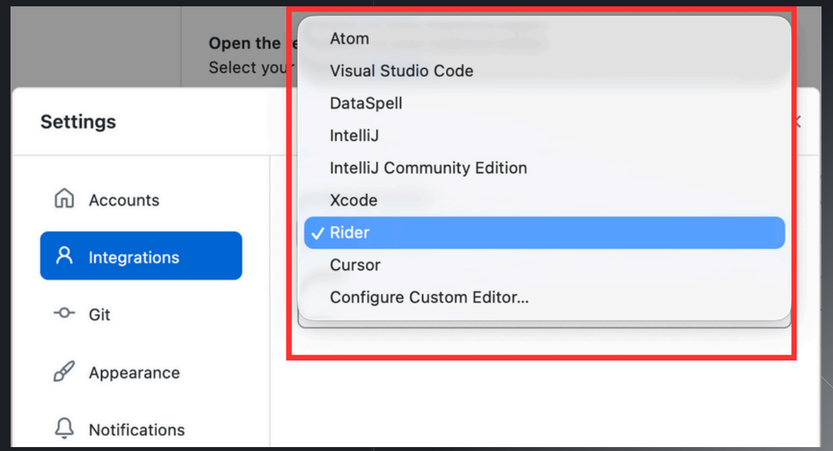


CHANGING YOUR EXTERNAL EDITOR

You can select your personally desired editor, by default it should be VS Code

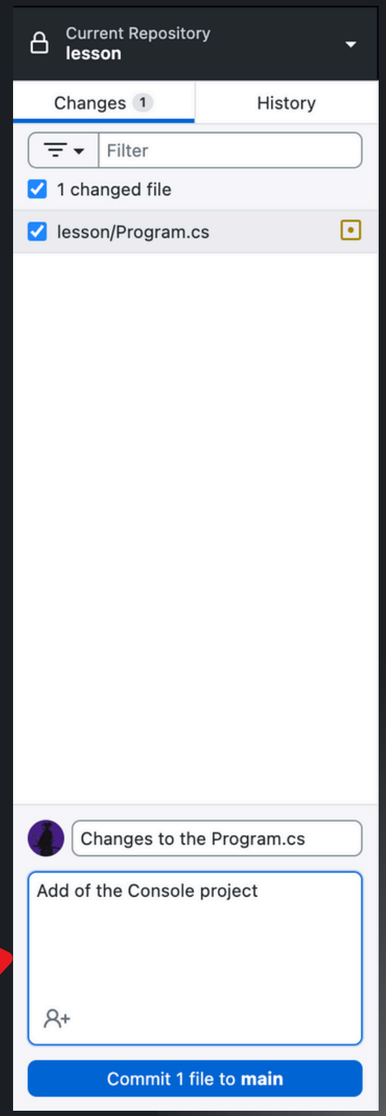






COMMITING CHANGES AND PUSHING TO GITHUB

Select the files which you want to add to commit

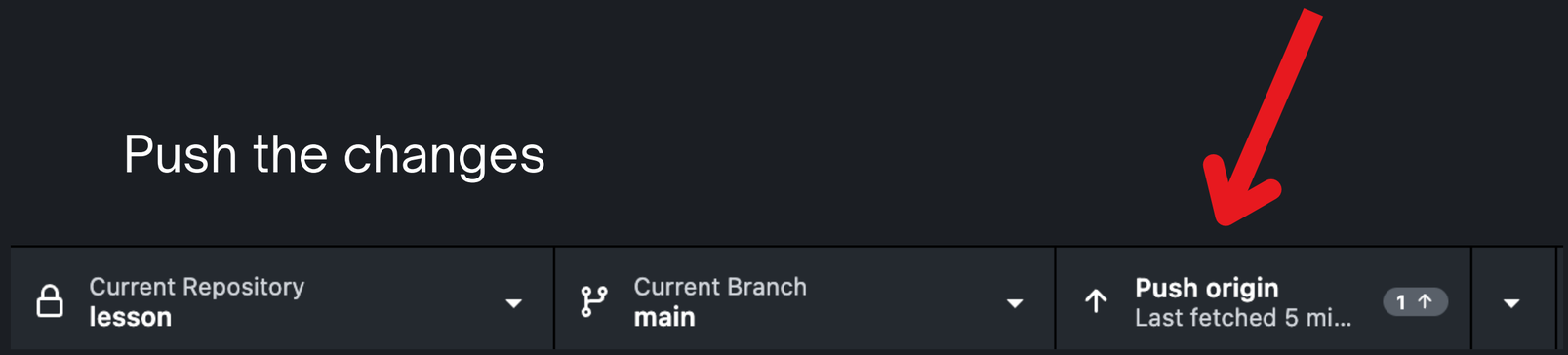


Add a Commit header

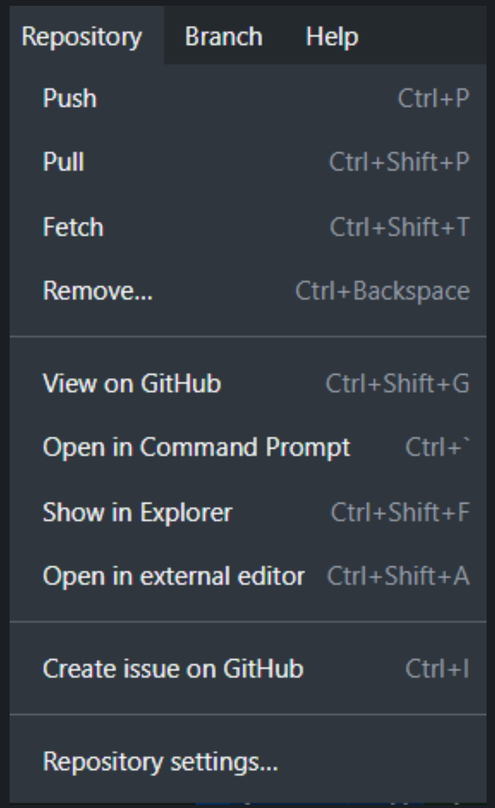
Add a Commit description

COMMITING CHANGES AND PUSHING TO GITHUB

Push the changes



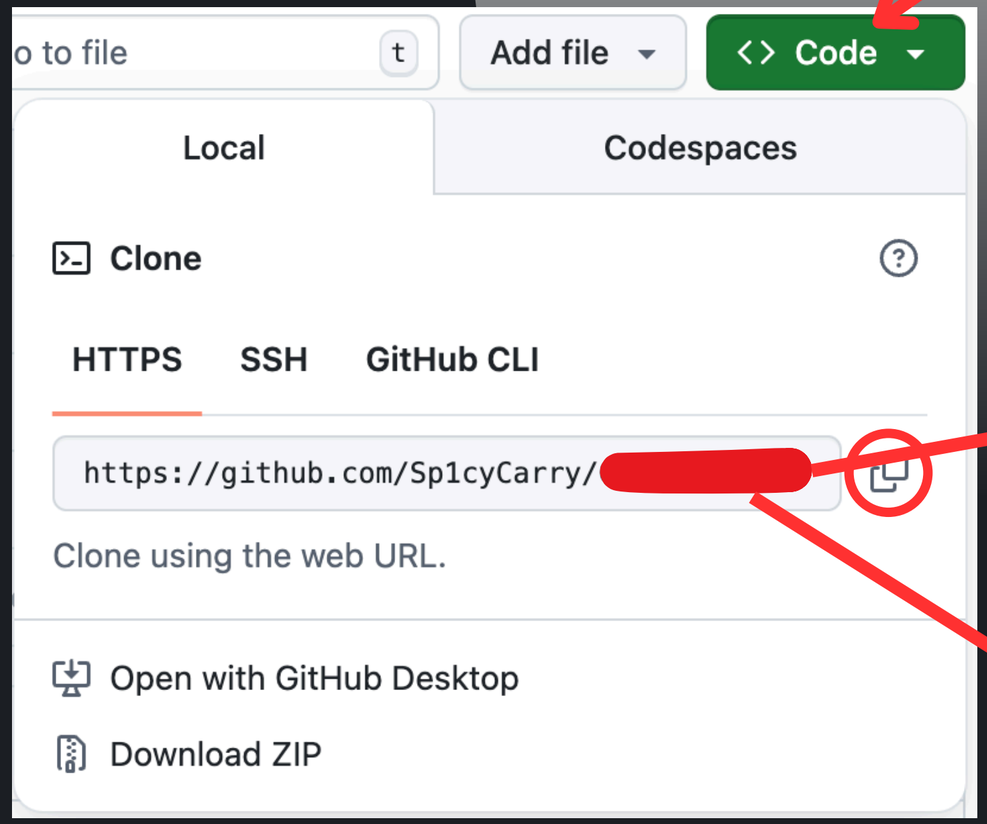
PULLING THE CHANGES



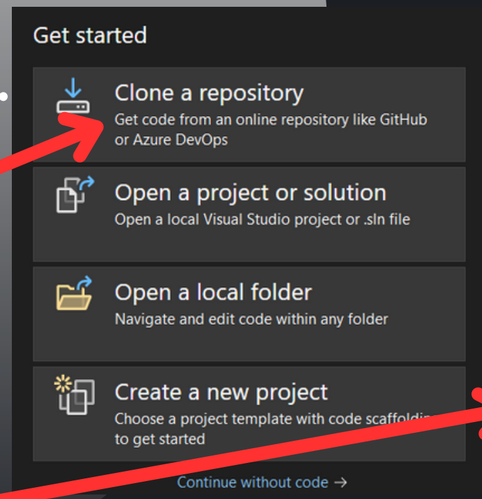
To pull the changes click on the pull button located in the Repository section.

CLONING REPO IN VISUAL STUDIO 2022

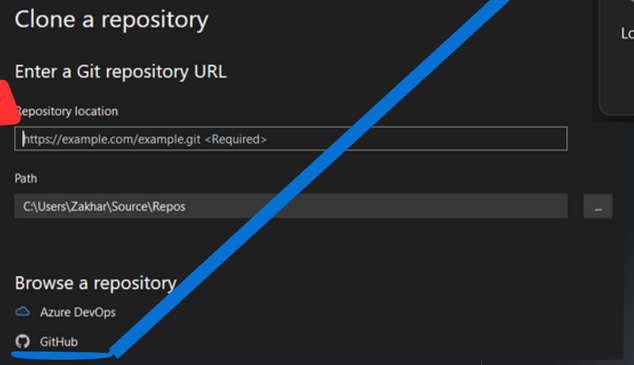
1.



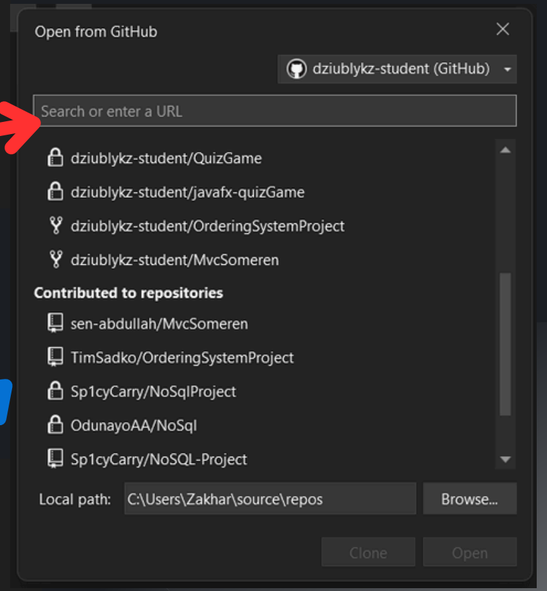
2.



3.

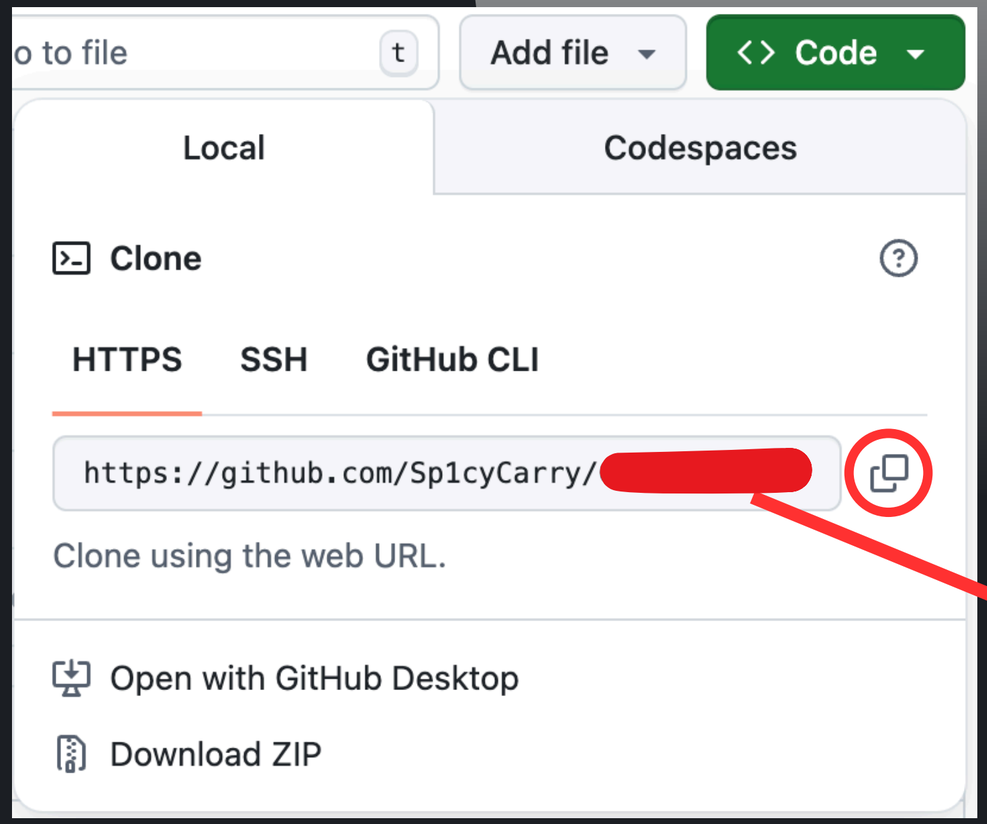


4.

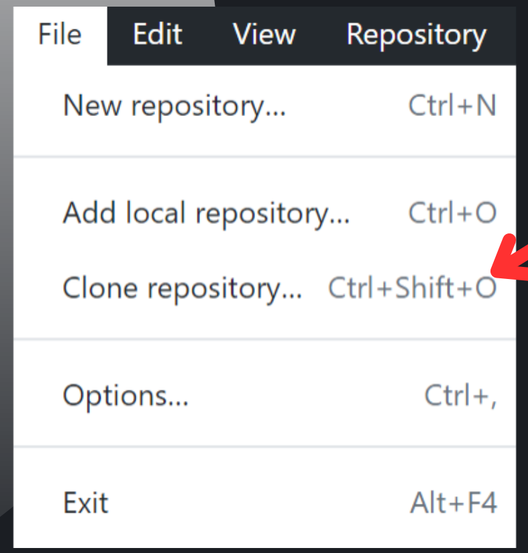


CLONING REPO WITH GITHUB DESKTOP

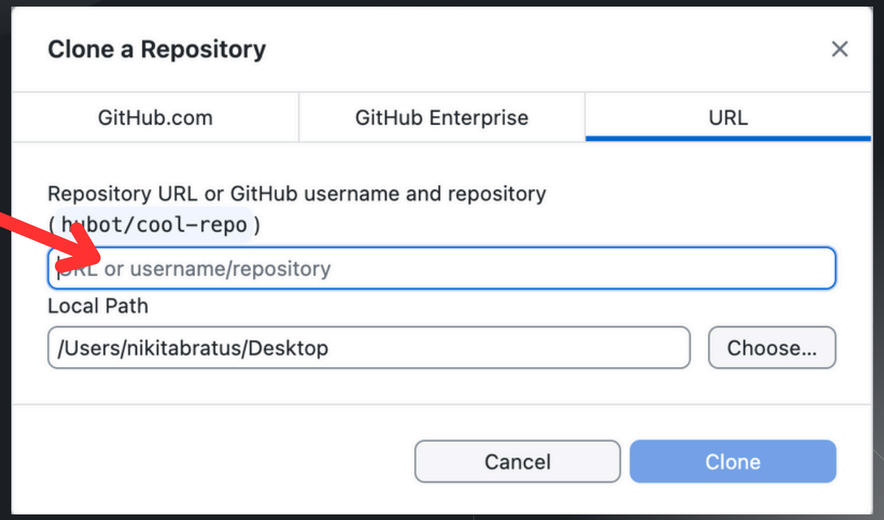
1.



2.



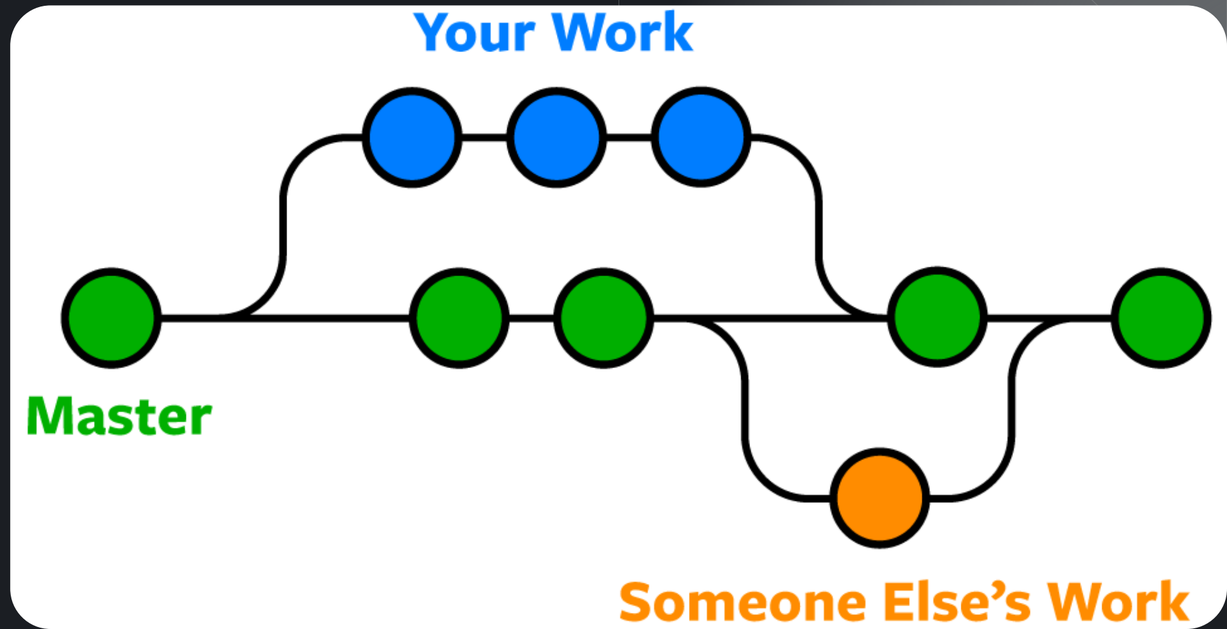
3.



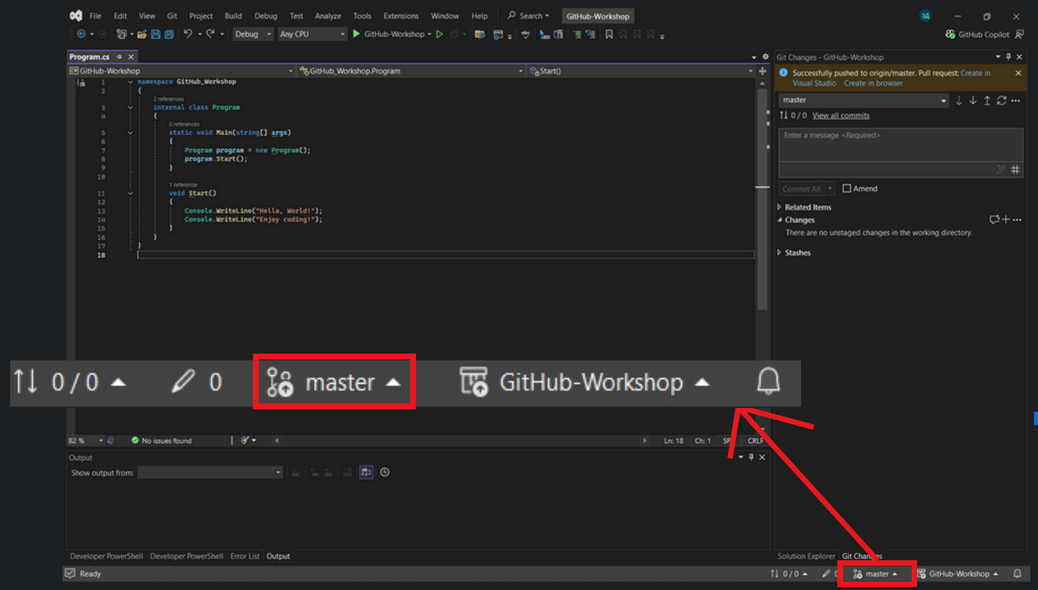
WHAT ARE BRANCHES?

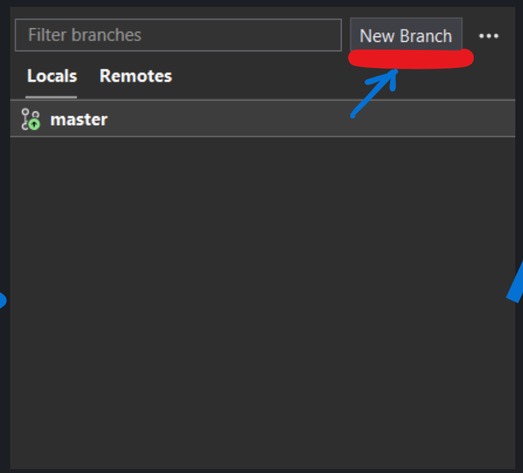
Branches prevent breaking the main codebase.

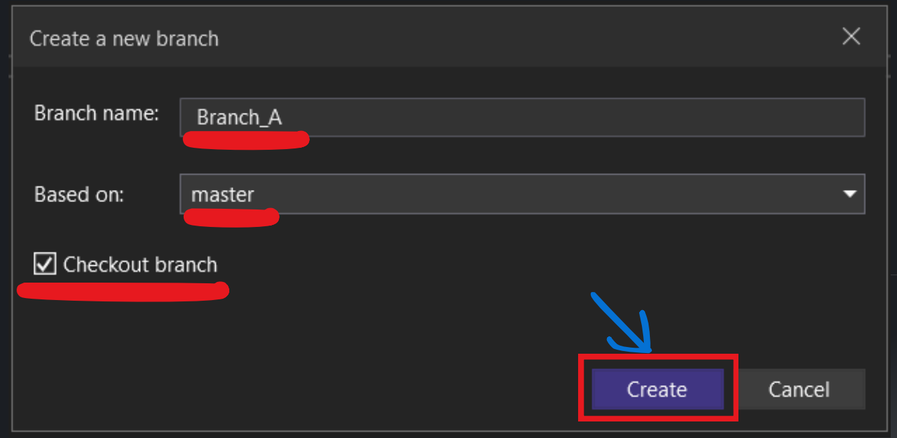
A branch is a separate line of development where you can build new features or fix bugs without touching the stable main branch.
Branches allow experimentation and prevent accidental damage to the project's primary version.



CREATING BRANCHES IN VISUAL STUDIO

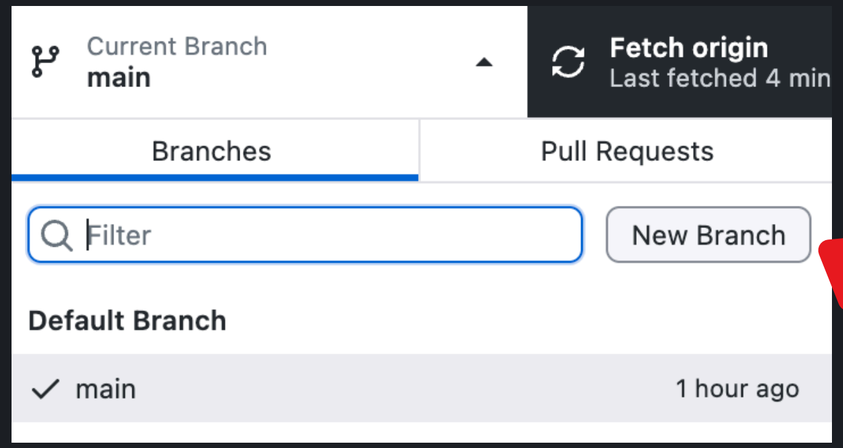




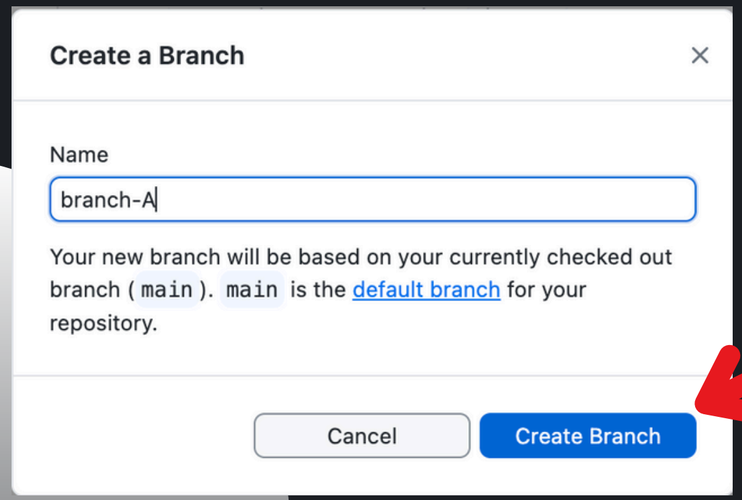


CREATING BRANCHES IN GITHUB DESKTOP

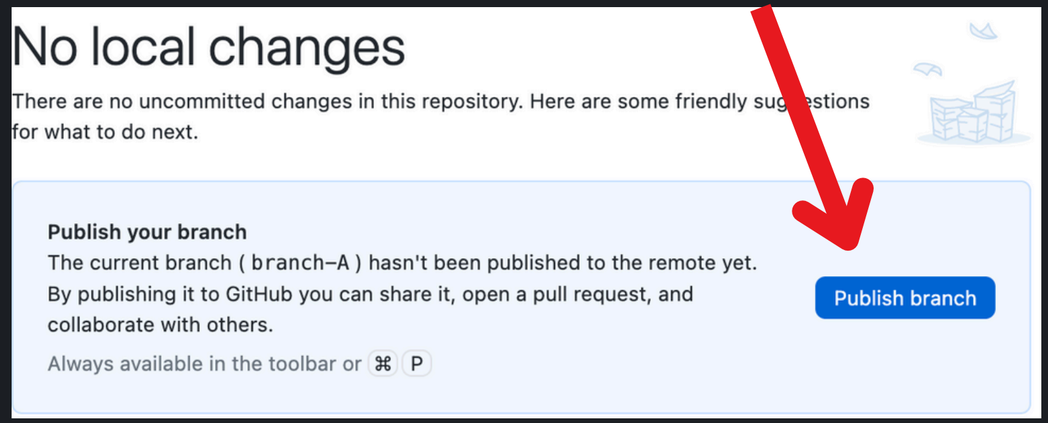
1.



2.

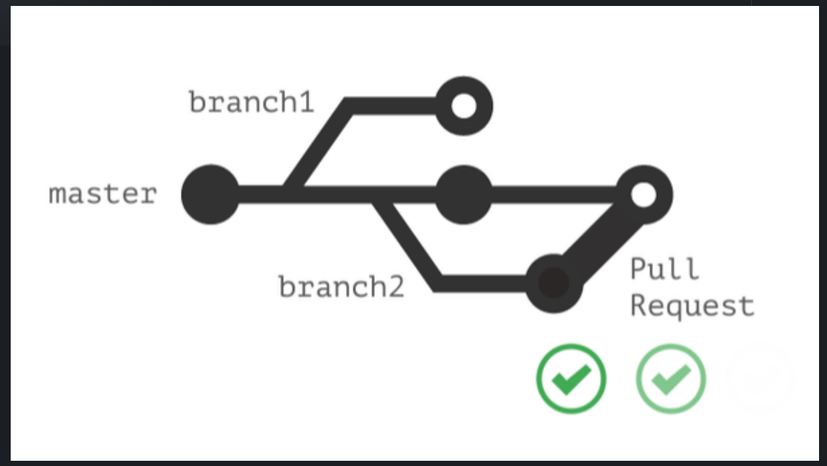


3.



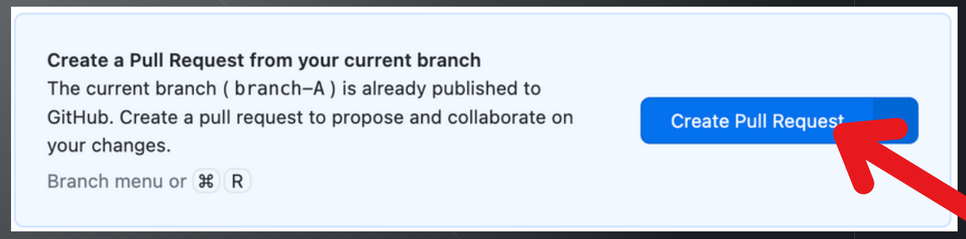
WHAT IS A PULL REQUEST?

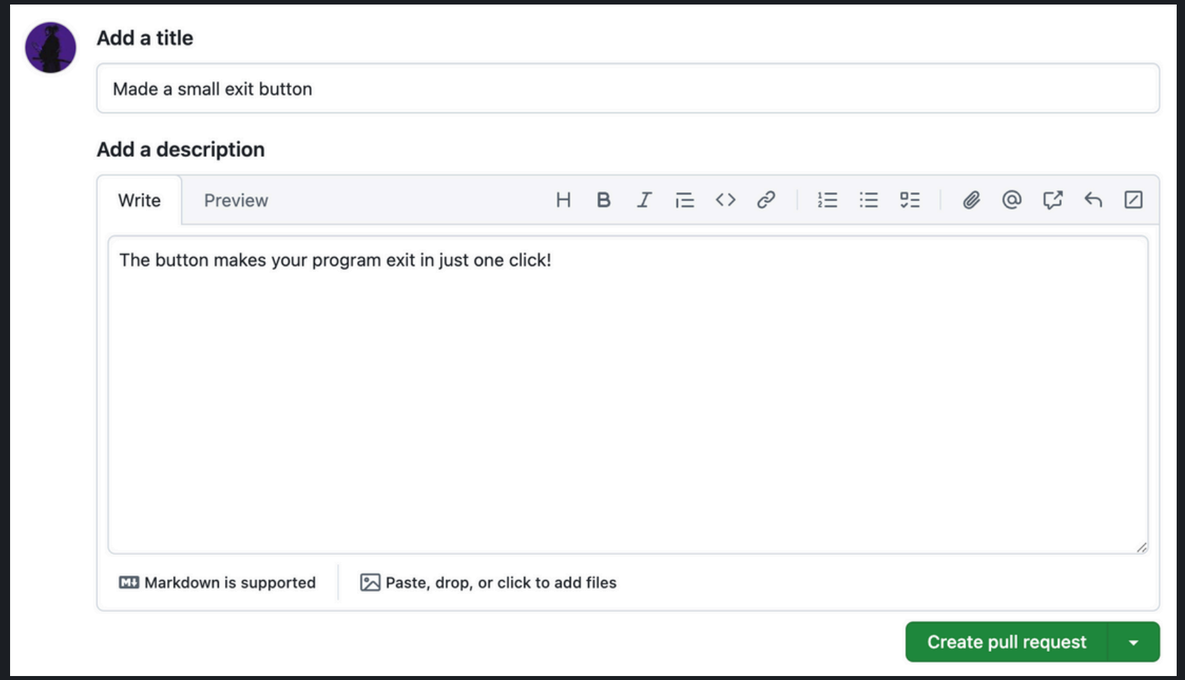
It lets your teammates see your changes, review the quality of the work, discuss improvements, and merge your contribution only when it’s ready.

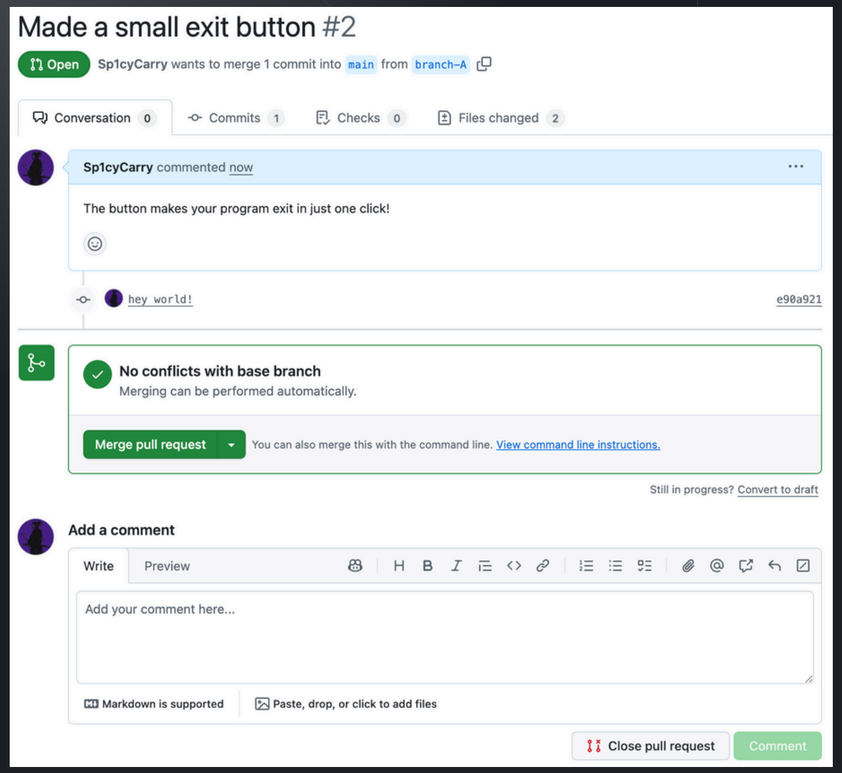


A **pull request** is how you **propose** your finished work to be merged into the main project.

PULL REQUEST PRACTICE







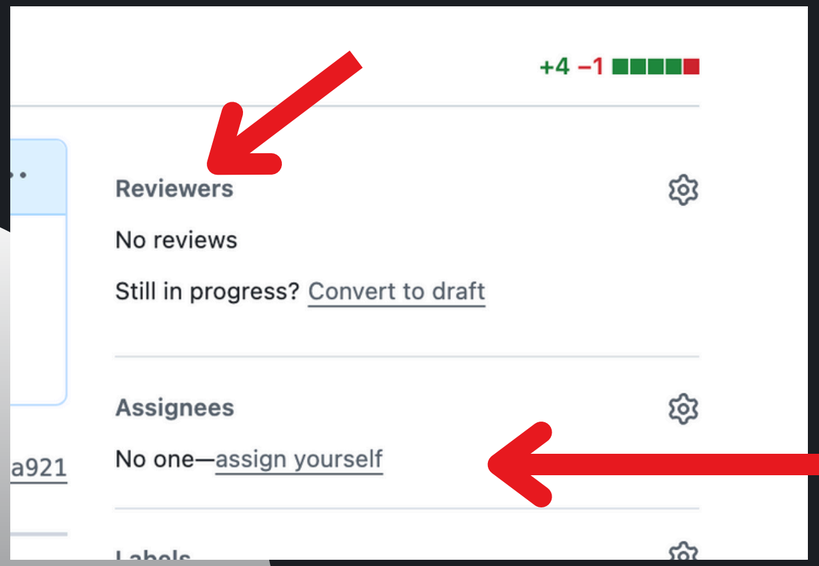
There is no major difference with pull requesting from GitHub Desktop or from Visual Studio

You could always make the pull request directly via GitHub and this will work in both scenarios

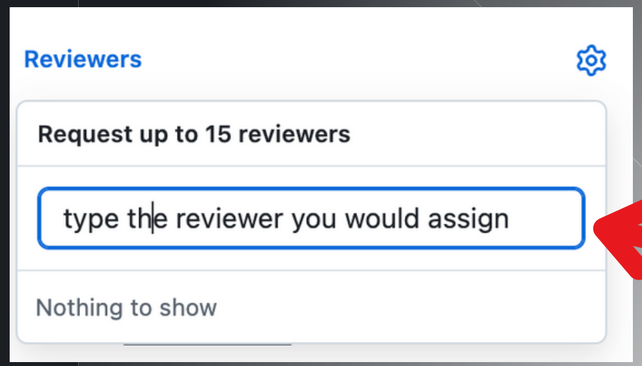
REVIEWER RESPONSIBILITIES

A reviewer looks at your changes carefully, checks for mistakes, gives feedback, and ensures the code fits the project’s goals.
Nothing should be merged into the main branch without a proper review, which helps maintain quality and prevent bugs.

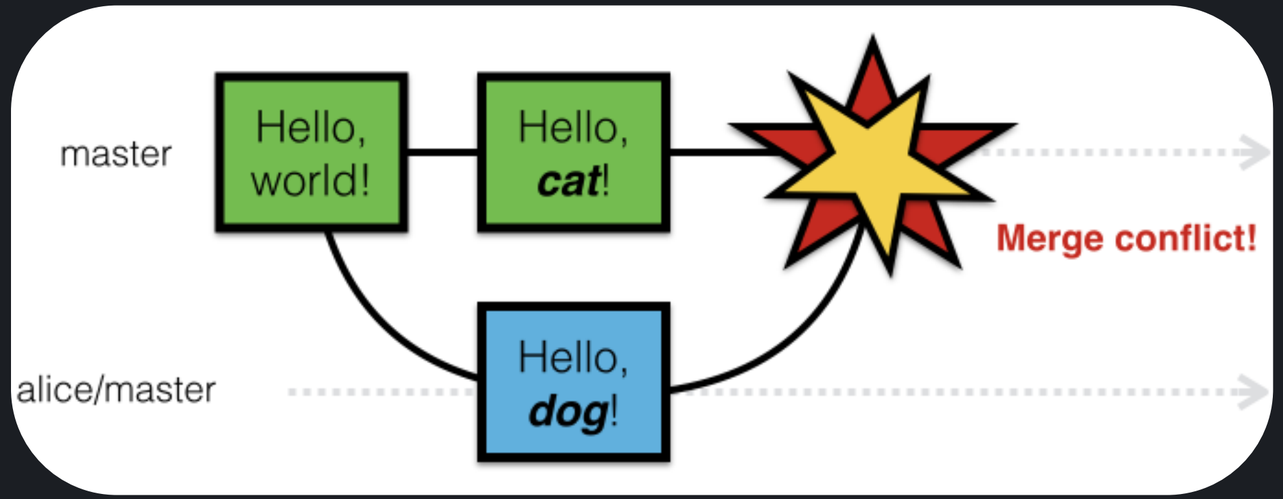
REVIEWER RESPONSIBILITIES IN ACTION



Assignees part is also handy to specify who did the exact feature

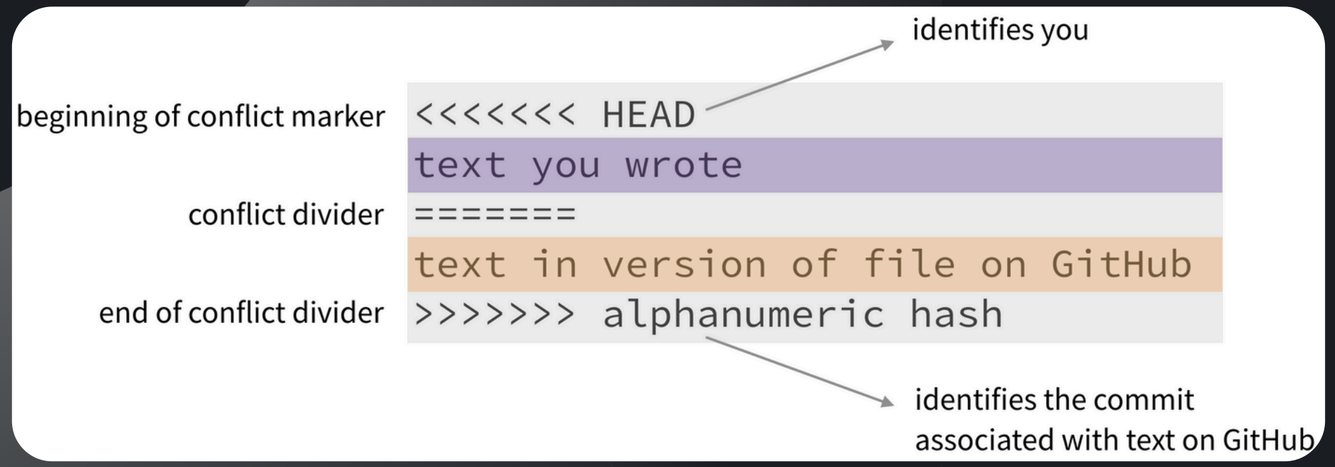


WHAT IS A MERGE CONFLICT?



A merge conflict happens when two people edit the same part of a file and Git cannot decide automatically which version to keep.
Instead of choosing incorrectly, Git asks you to manually resolve the conflict using clear conflict markers that show both versions of the code.

Git can’t decide whose version is correct

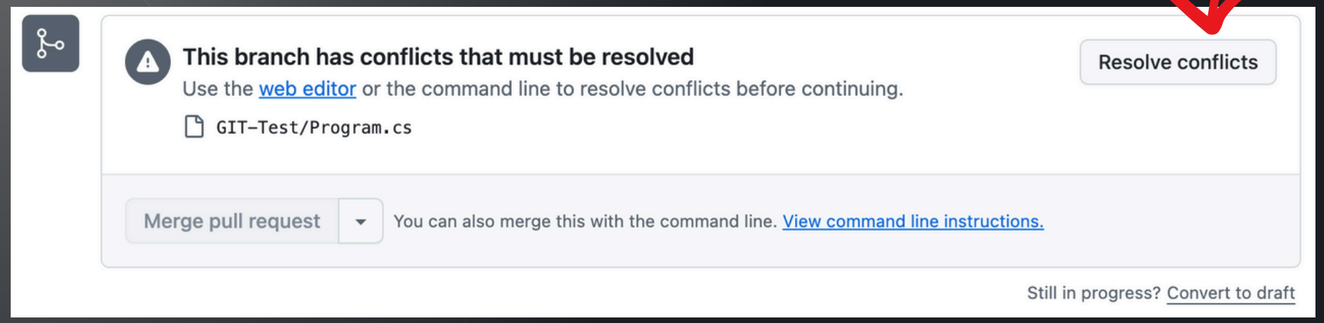


HOW TO SOLVE MERGE CONFLICTS

This process is easier when done with your teammate.

To resolve a conflict, you open the file, compare the two versions that Git shows you, decide which lines belong in the final result, remove Git's conflict markers, and confirm that the issue is resolved.

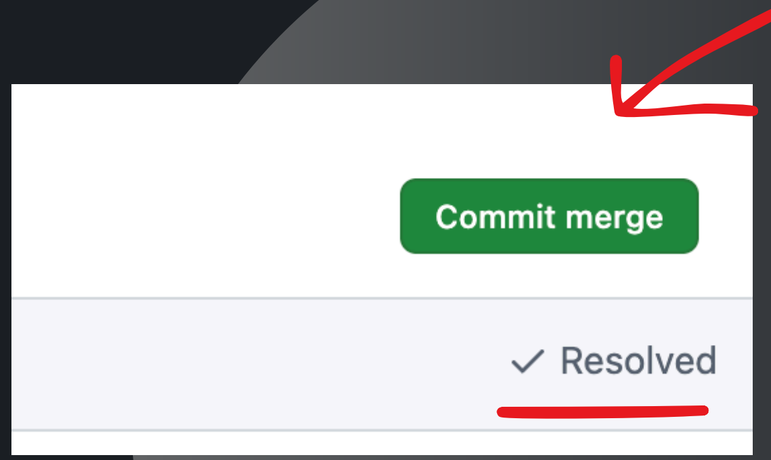
HOW TO SOLVE MERGE CONFLICTS



1. Person A creates Branch A, changes a line, commits, pushes, and merges it into main (auto-merge).
2. Person B creates Branch B (without pulling), changes the same line, commits, pushes, and opens a Pull Request.
3. GitHub shows “Can’t automatically merge” → click Resolve conflicts.
4. In the conflict editor, choose “Accept current change”.
5. Click Mark as resolved, then Commit merge and finish the Pull Request.

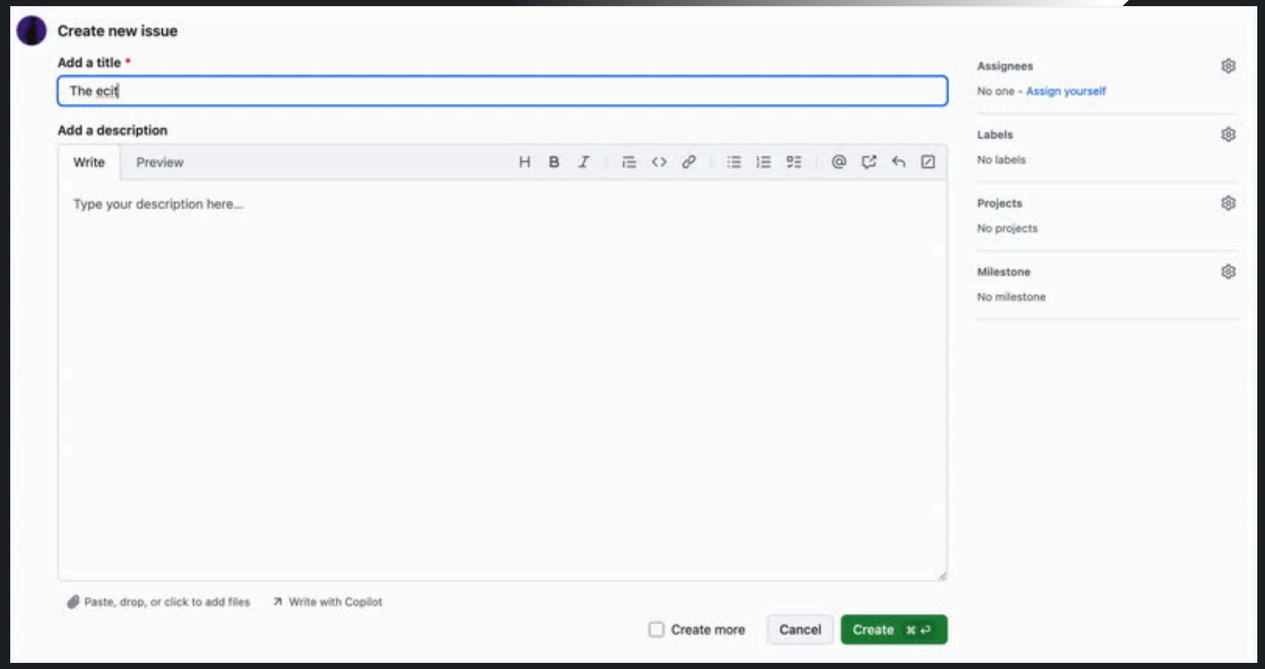
GIT-Test/Program.cs

```
1   namespace GIT_Test;
2
3   class Program
4   {
5       static void Main(string[] args)
6       {
    Accept current change | Accept incoming change | Accept both changes
7   <<<<<<< zakhar-merge (Current change)
8           Console.WriteLine("Hello from Branch Zakhar!");
9   =======
10          Console.WriteLine("Hello from Branch Nikita!");
11  >>>>>>> main (Incoming change)
12      }
13  }
```



USING GITHUB ISSUES

GitHub Issues provide a simple way to plan and organize tasks.
You can track bugs, assign responsibilities, label tasks by type, and even connect issues directly to your pull requests as you complete features.



GIT PROJECTS

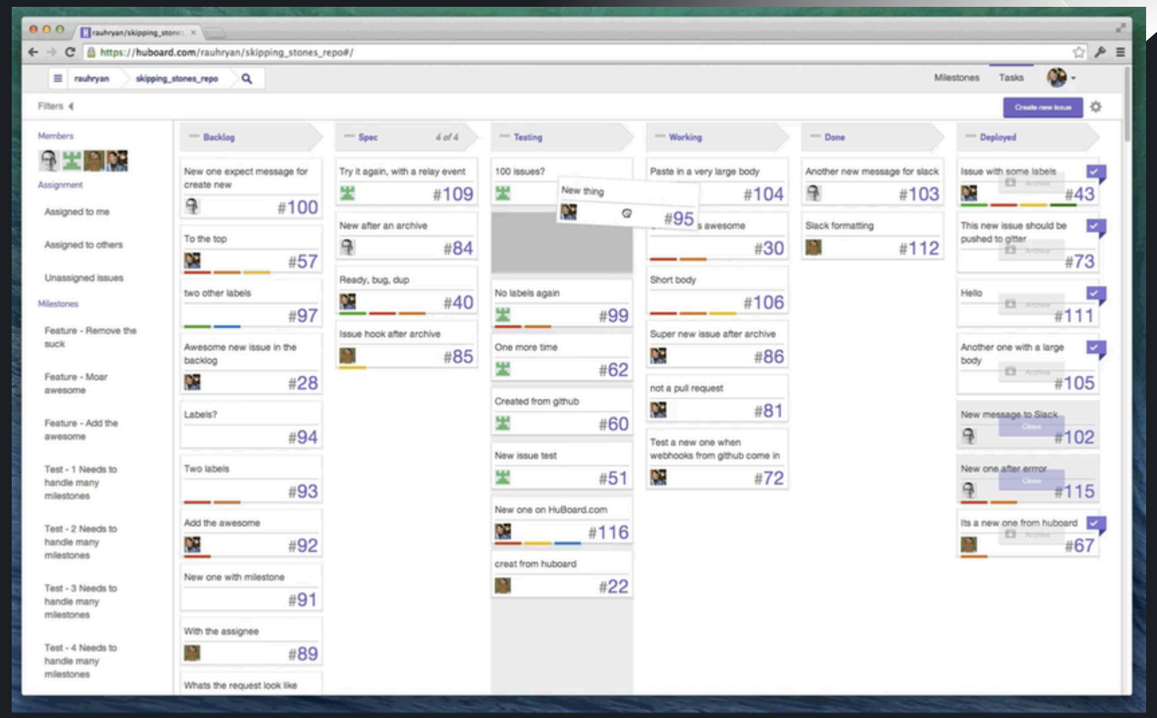
GitHub Projects help you plan, track, and manage tasks connected to your repository.

- Works like a task board (Kanban)
- Uses columns: To Do / In Progress / Done
- Can be linked to Issues and Pull Requests

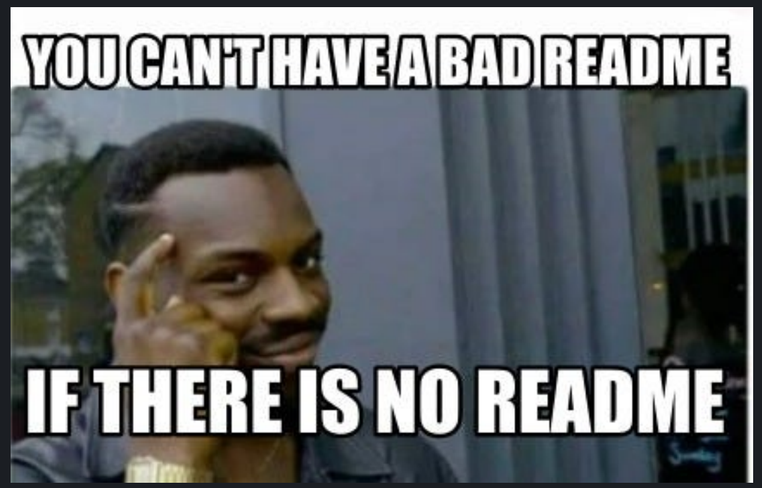
Think of it as: A project management board for your code

In GitHub:

- Projects → New Project → Add issues/cards



WRITING A GOOD README



A README introduces your project to the world.

It should describe what the project does, how to run it, which tools or libraries you used, any screenshots or examples, and the names of the contributors.

GIT BEST PRACTICES

Good Git habits keep your team productive: commit often, always pull before pushing, use branches for new features, write meaningful commit messages, and keep the main branch clean and stable.

These habits prevent confusion, conflicts, and unnecessary errors.

Thank **You**

Nikita Bratus
736115@student.inholland.nl

-

Zakhar Dziublyk
731522@student. inholland.nl