



Introduction to GitHub

Next Slide

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



TODAY'S MENU

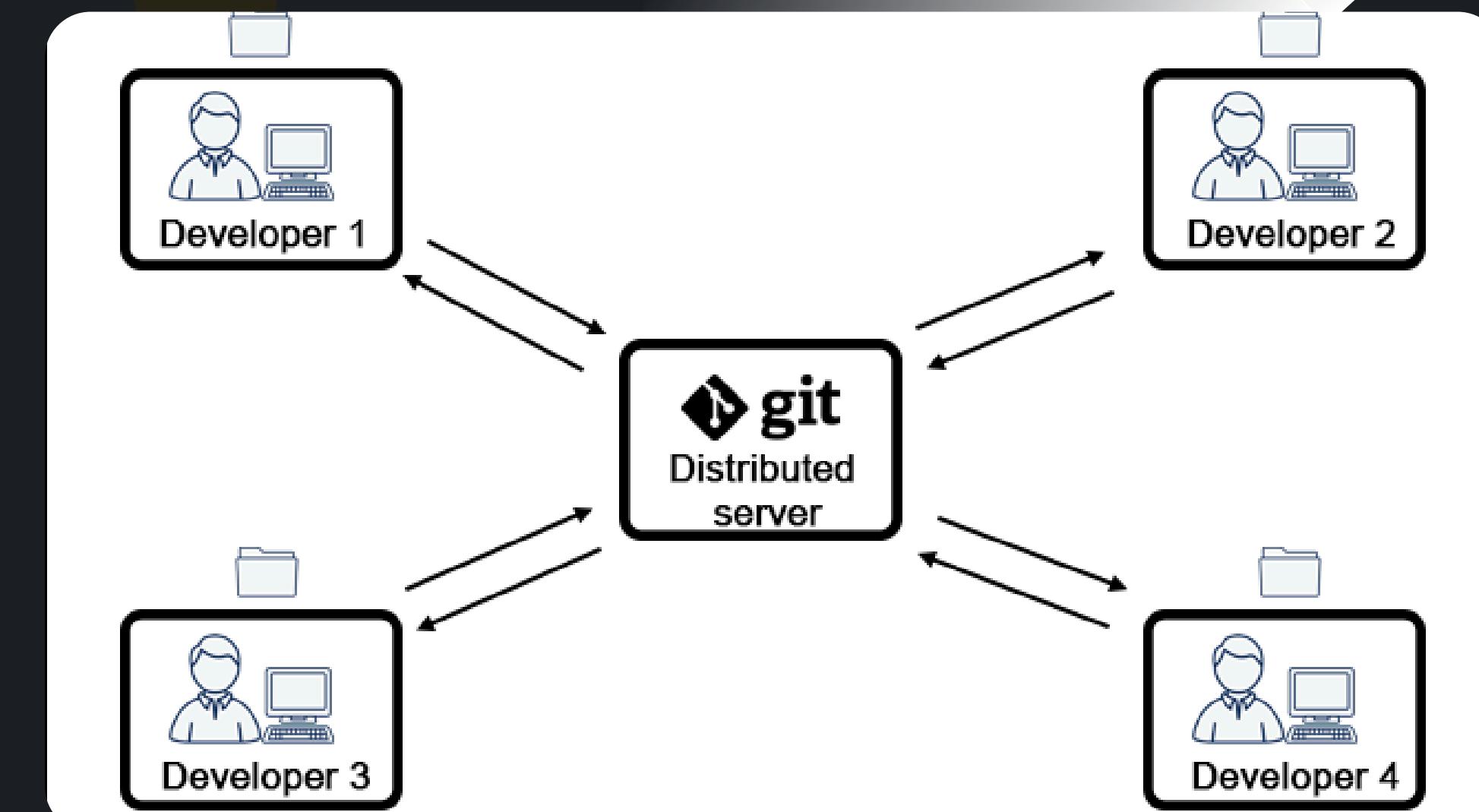
- What is Git?
- What is GitHub?
- Local vs Remote and basic Git operations
- Visual studio & GitHub Desktop
- Practice basic operations
- .gitignore file
- What are branches?
- Practice creating branches
- What is a pull request?
- Practice creating pull request
- Reviewer responsibilities
- What is 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?

A large, semi-transparent watermark of the GitHub logo is visible in the background, consisting of the word "GitHub" in a stylized font with a cat silhouette integrated into the letter "i".

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

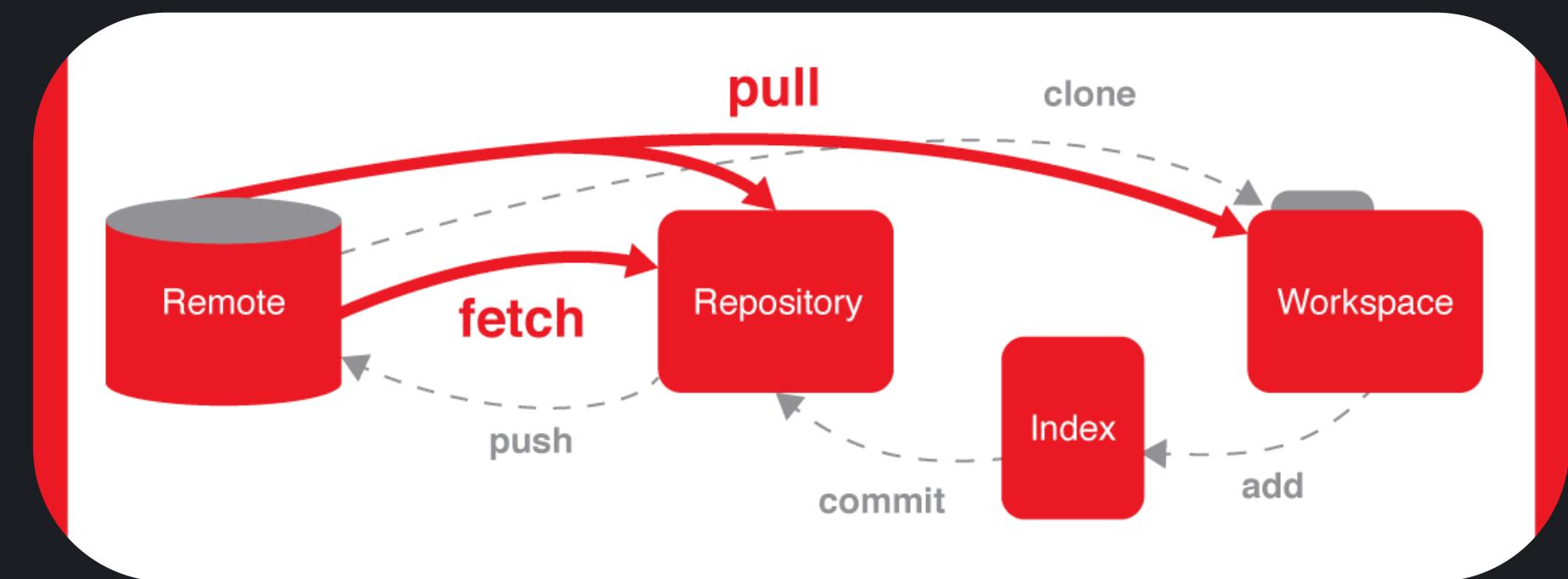


LOCAL VS REMOTE

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

The screenshot shows a GitHub repository named "octo-repo" which is private. The main branch is selected. The commit history shows two recent commits:

- A commit from "octocat" updating the README.md file, made 2 minutes ago with 178 commits.
- A commit from ".github" updating codeql-analysis.yml, made 13 minutes ago.



INDEXING IN SIMPLE WORDS

The screenshot shows a GitHub commit history. At the top, there are navigation buttons: 'Go to file' (with a search icon), a dropdown menu, and a green 'Code' button. Below these are five commits listed vertically. The first commit, 'Hi from Branch A!', has a red box around its author, timestamp ('3 days ago'), and '4 Commits' link. The other four commits are simple 'initial commit' entries with timestamps of '3 days ago'. The bottom commit, 'Create README.md', also has a timestamp of '3 days ago'.

Commit Message	Time Ago
Hi from Branch A!	3 days ago
initial commit	3 days ago
initial commit	3 days ago
Create README.md	3 days ago

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

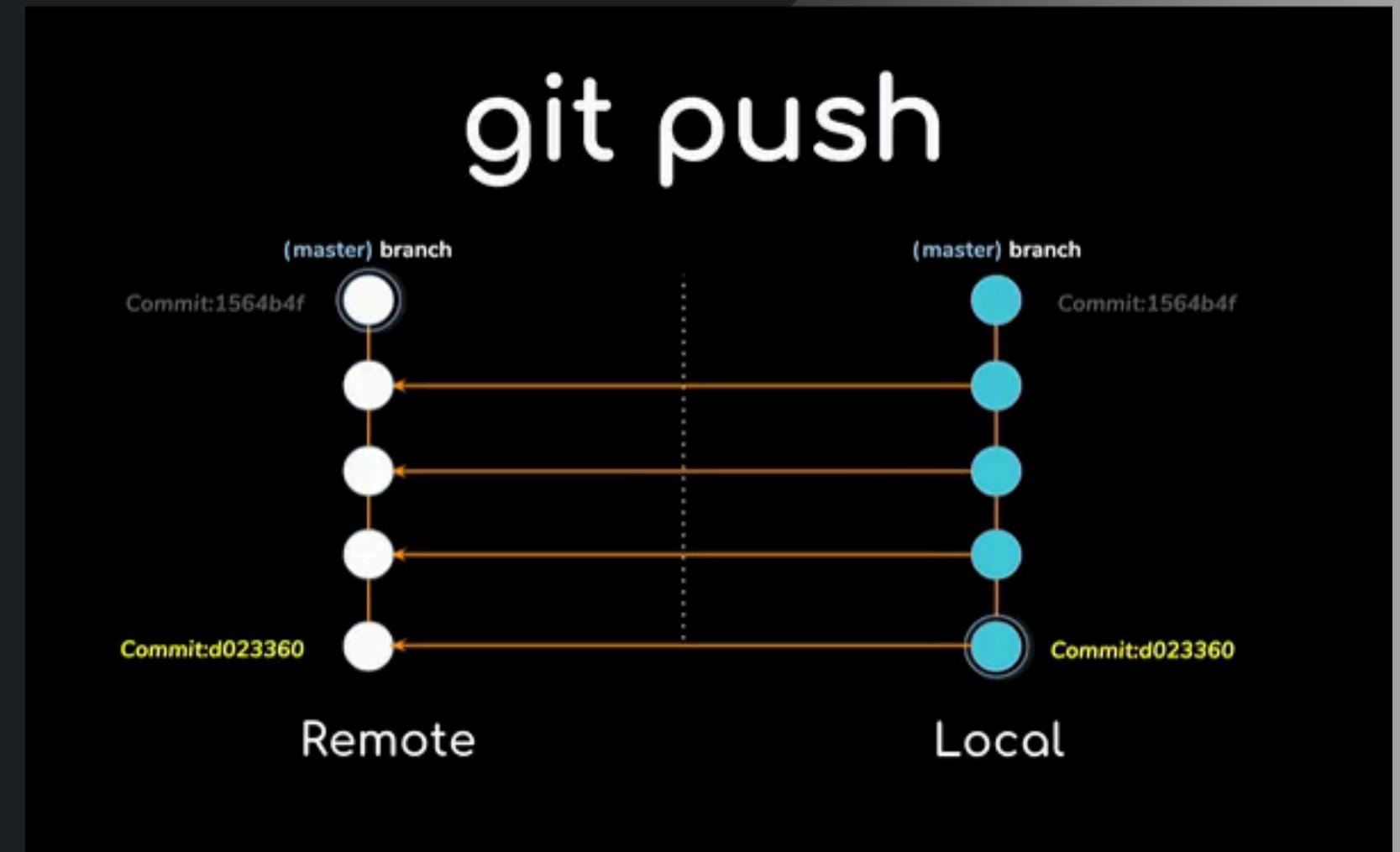


PUSH PROCESS

Sharing Your Work with the Team

Push sends your local commits to the remote GitHub repository.

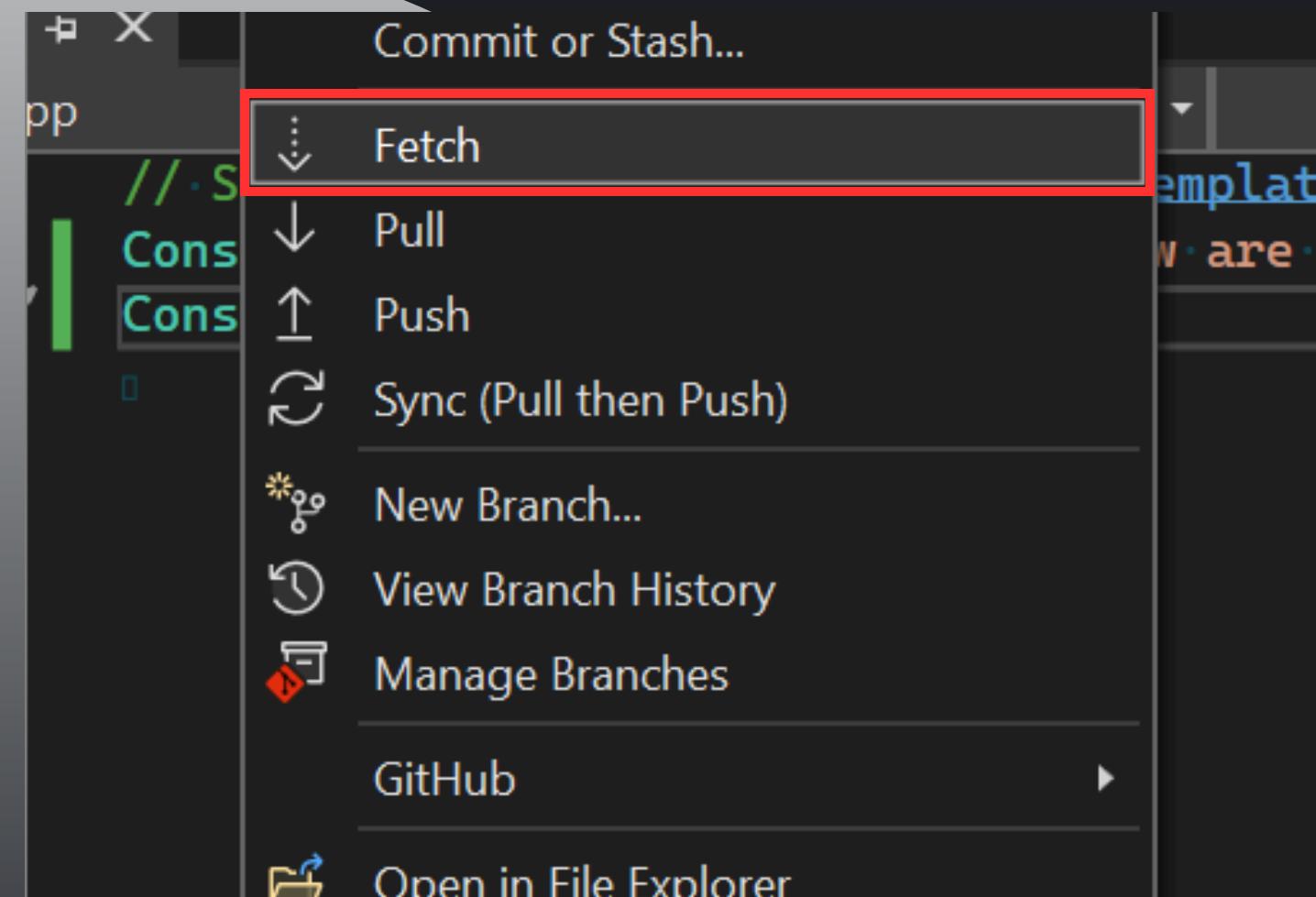
- Uploads changes to GitHub
- Makes them visible to others
- Might be blocked on protected branches





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

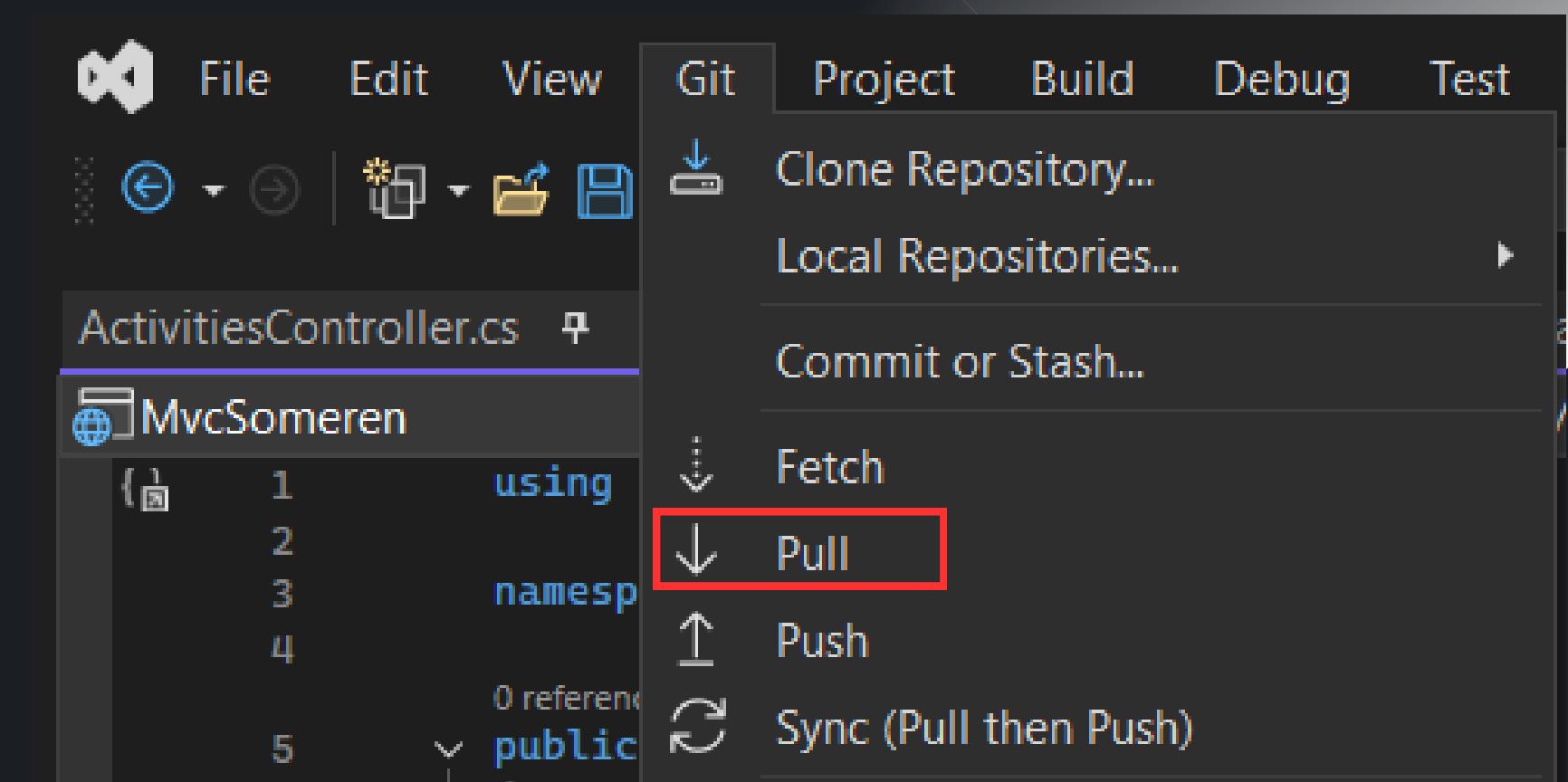
Updating Your Local Project



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”

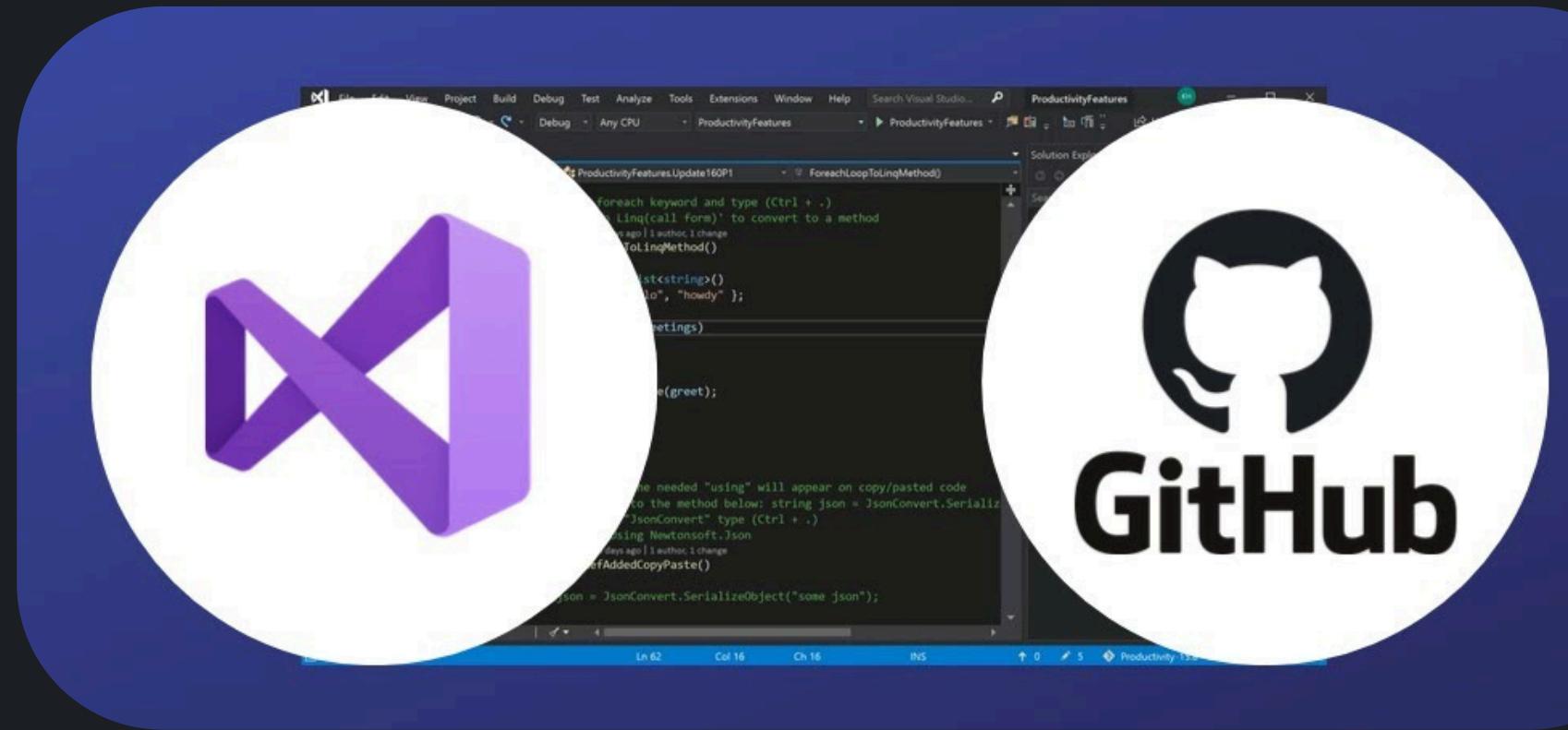




VISUAL STUDIO + GIT

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.

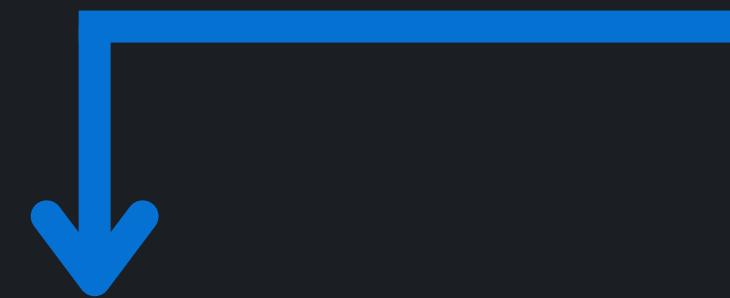


Visual Studio provides built-in Git tools:

- “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



Additional information

Console App C# Linux macOS Windows Console

Framework .NET 8.0 (Long Term Support)

Enable container support

Container OS Linux

Container build type Dockerfile

Do not use top-level statements

Enable native AOT publish

1. Configure your new project

Console App C# Linux macOS Windows Console

Project name GitHub-Workshop

Location GitHub\Git-2.0

Solution name GitHub-Workshop

Place solution and project in the same directory

Project will be created in GitHub\Git-2.0\GitHub-Workshop\

2.

3.

File Edit View Git Debug Analyze Tools Extensions

Clone Repository...

Create Git Repository...

Local Repositories...

Commit or Stash



SETTING UP A GIT REPOSITORY

5.

```
namespace GitHub_Workshop
{
    internal class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Hello, World!");
        }
    }
}
```

4.

Create a Git repository

Push to a new remote

1 Initialize a local Git repository

1 Local path

.gitignore template Default (VisualStudio)

License template None

Add a README.md

Other

2 Existing remote

3 Local only

Create a new GitHub repository

2 Account

3 Owner

Repository name GitHub-Workshop

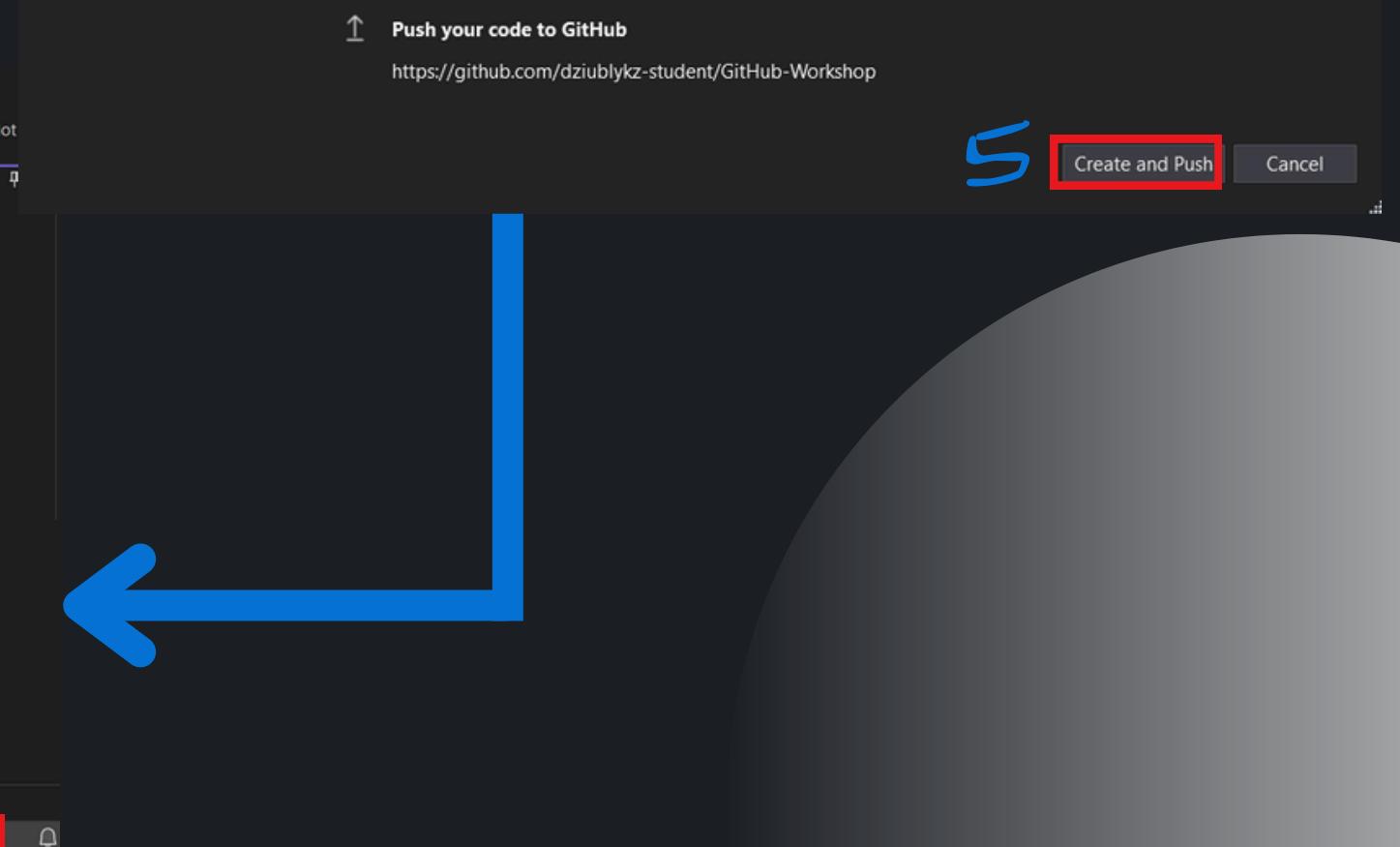
Description Enter the description of the GitHub repository <Optional>

4 Visibility Private

You choose who can see and commit to this repository.

Push your code to GitHub

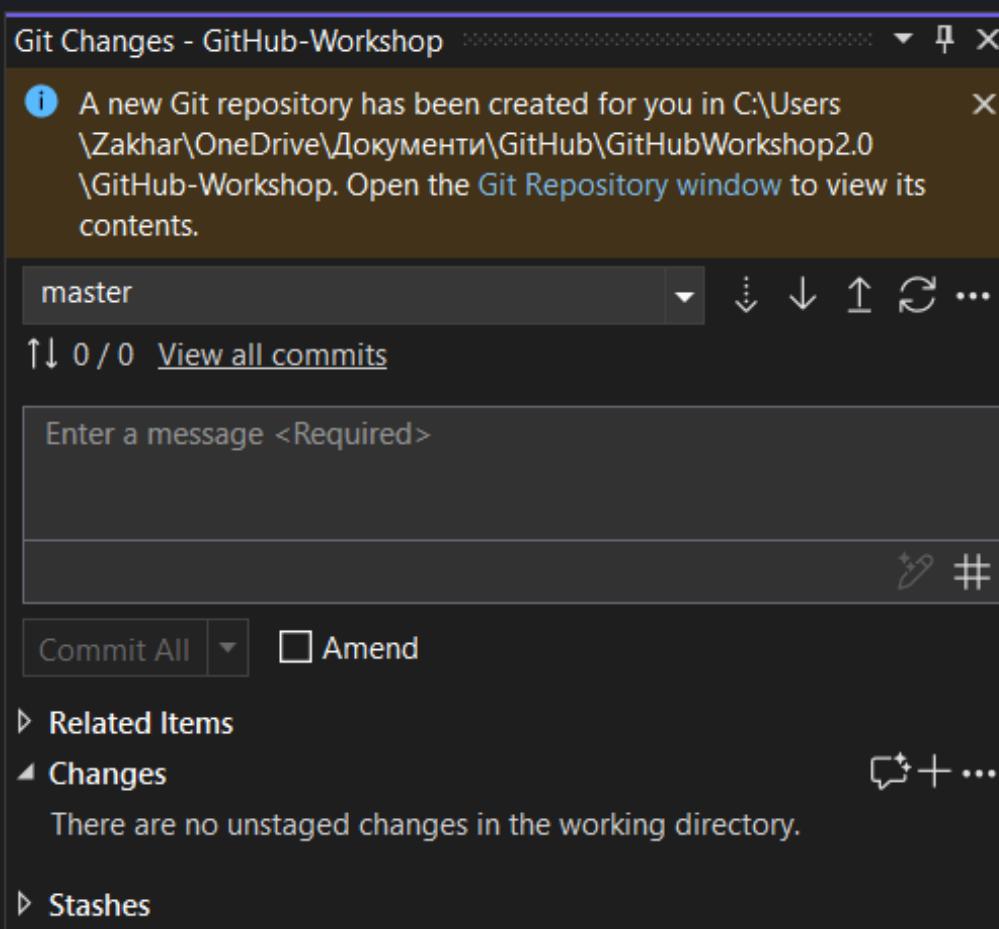
<https://github.com/dziublykz-student/GitHub-Workshop>



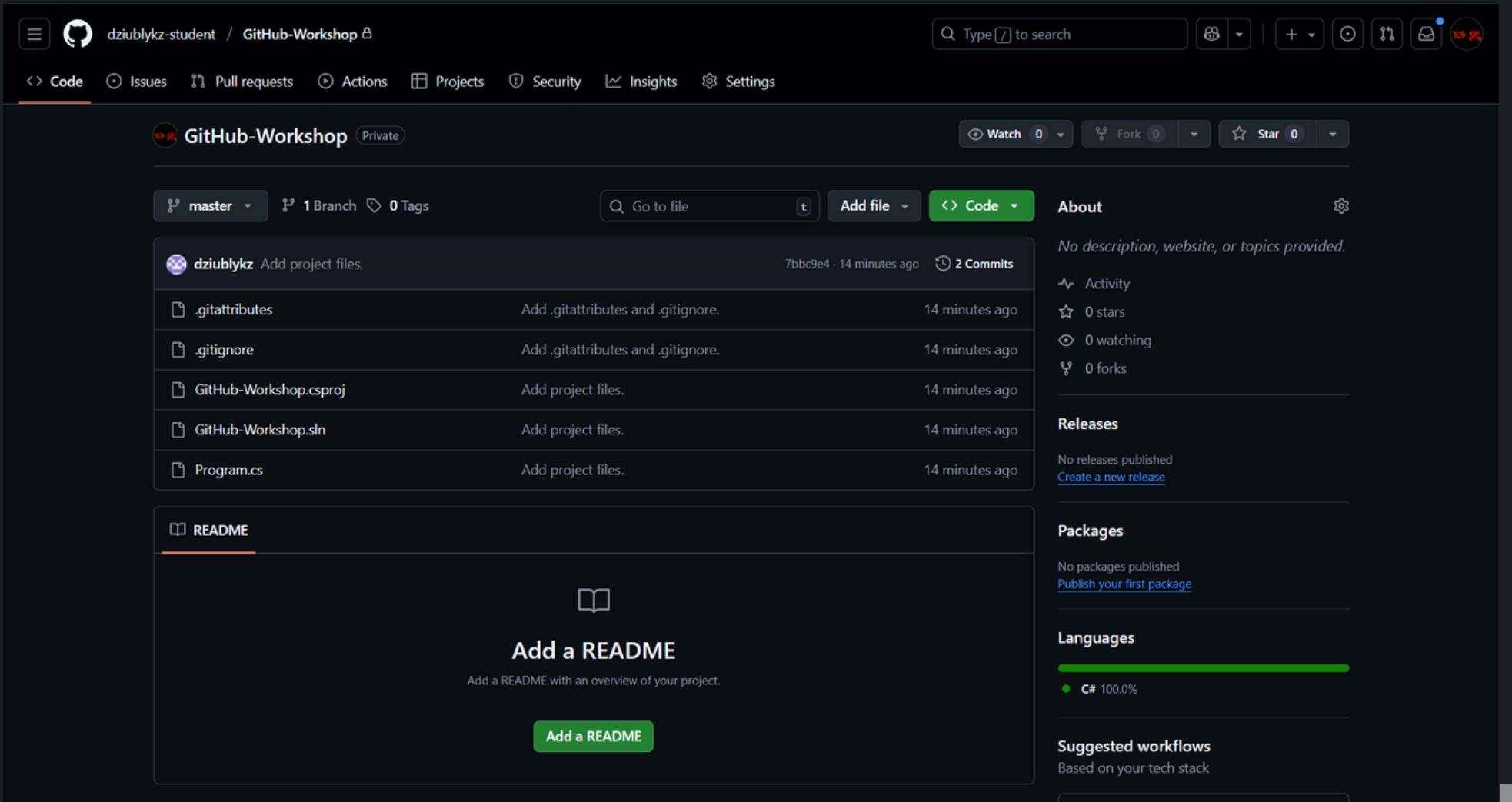


SETTING UP A GIT REPOSITORY

6.



7.





1.

The screenshot shows the GitHub repository settings page for 'GitHub-Workshop'. The 'Settings' tab is active. In the sidebar, the 'Access' section is expanded, showing the 'Collaborators' option, which is highlighted with a red box. A blue arrow points from the top of the image towards this red box.

2.

The screenshot shows the GitHub repository settings page for 'GitHub-Workshop'. The 'General' tab is active. In the sidebar, the 'Access' section is expanded, showing the 'Collaborators' option, which is highlighted with a red box. A blue arrow points from the previous screenshot towards this red box.

3.

The screenshot shows the GitHub repository settings page for 'GitHub-Workshop'. The 'General' tab is active. In the main area, the 'Collaborators and teams' section is shown. It displays a message: 'You haven't invited any collaborators yet'. Below this message is a red button labeled 'Add people', which is highlighted with a red box. A blue arrow points from the bottom of the previous screenshot towards this red box.

COLLABORATORS



4.

Add people to GitHub-Workshop

Search by username, full name, or email

Q nikita

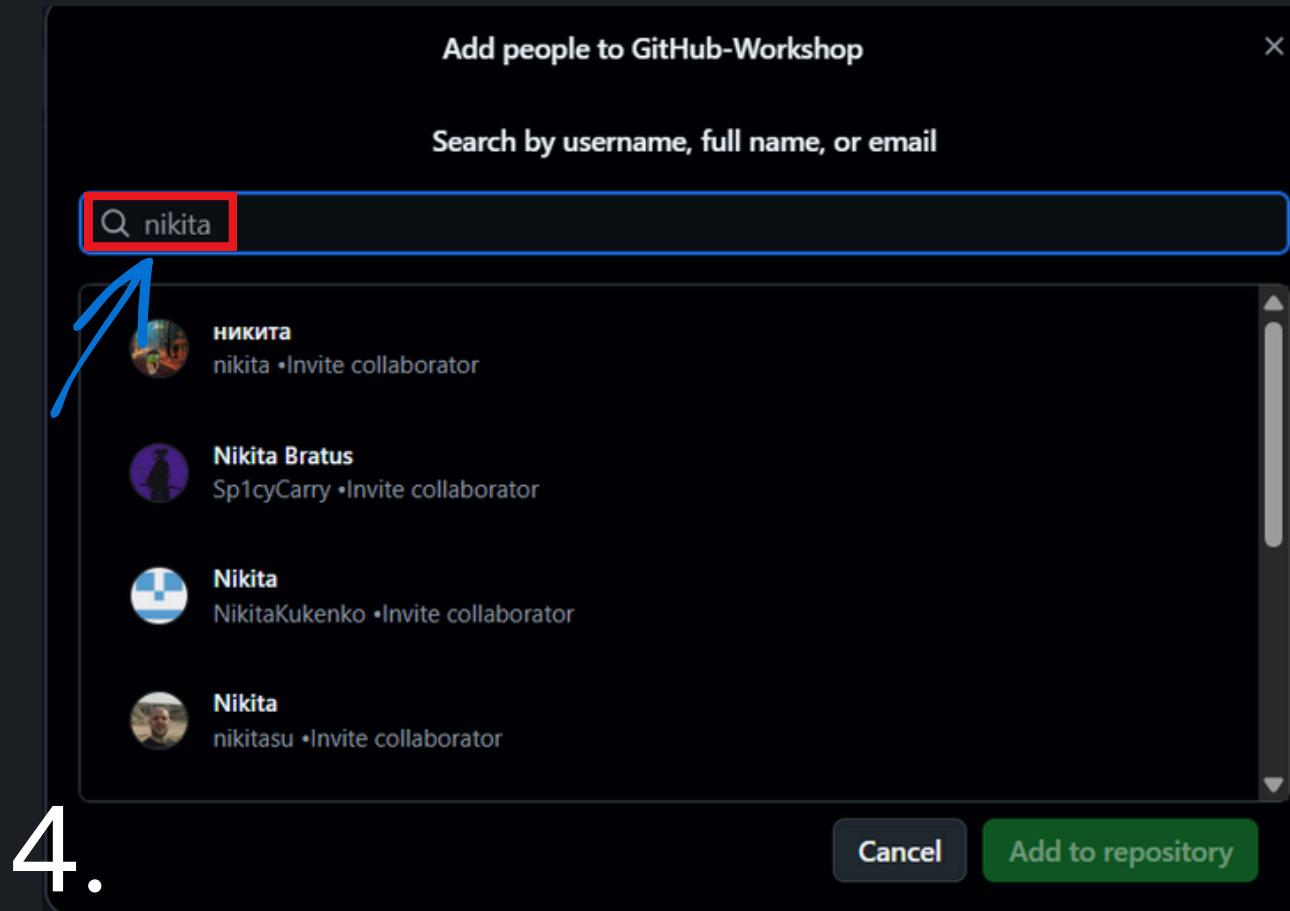
никита
nikita •Invite collaborator

Nikita Bratus
Sp1cyCarry •Invite collaborator

Nikita
NikitaKukenko •Invite collaborator

Nikita
nikitasu •Invite collaborator

Cancel Add to repository

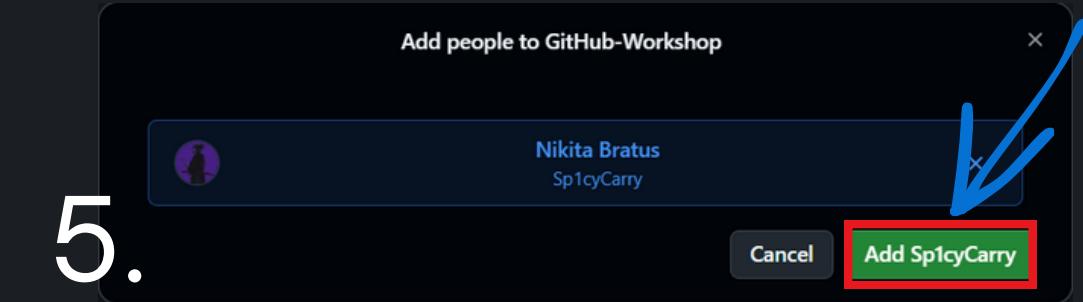


5.

Add people to GitHub-Workshop

Nikita Bratus
Sp1cyCarry

Cancel Add Sp1cyCarry



6.

dziublykz-student / GitHub-Workshop

Type to search

Code Issues Pull requests Actions Projects Security Insights Settings

Sp1cyCarry has been added as a collaborator on the repository.

General Collaborators and teams

Access

Collaborators

Private repository Only those with access to this repository can view it

Manage visibility

Code and automation

Branches

Tags

Rules

Actions

Models

Webhooks

Copilot

Environments

Codespaces

Pages

Direct access 1 entity has access to this repository: [0 collaborators. 1 invitation.](#)

Manage access

Add people

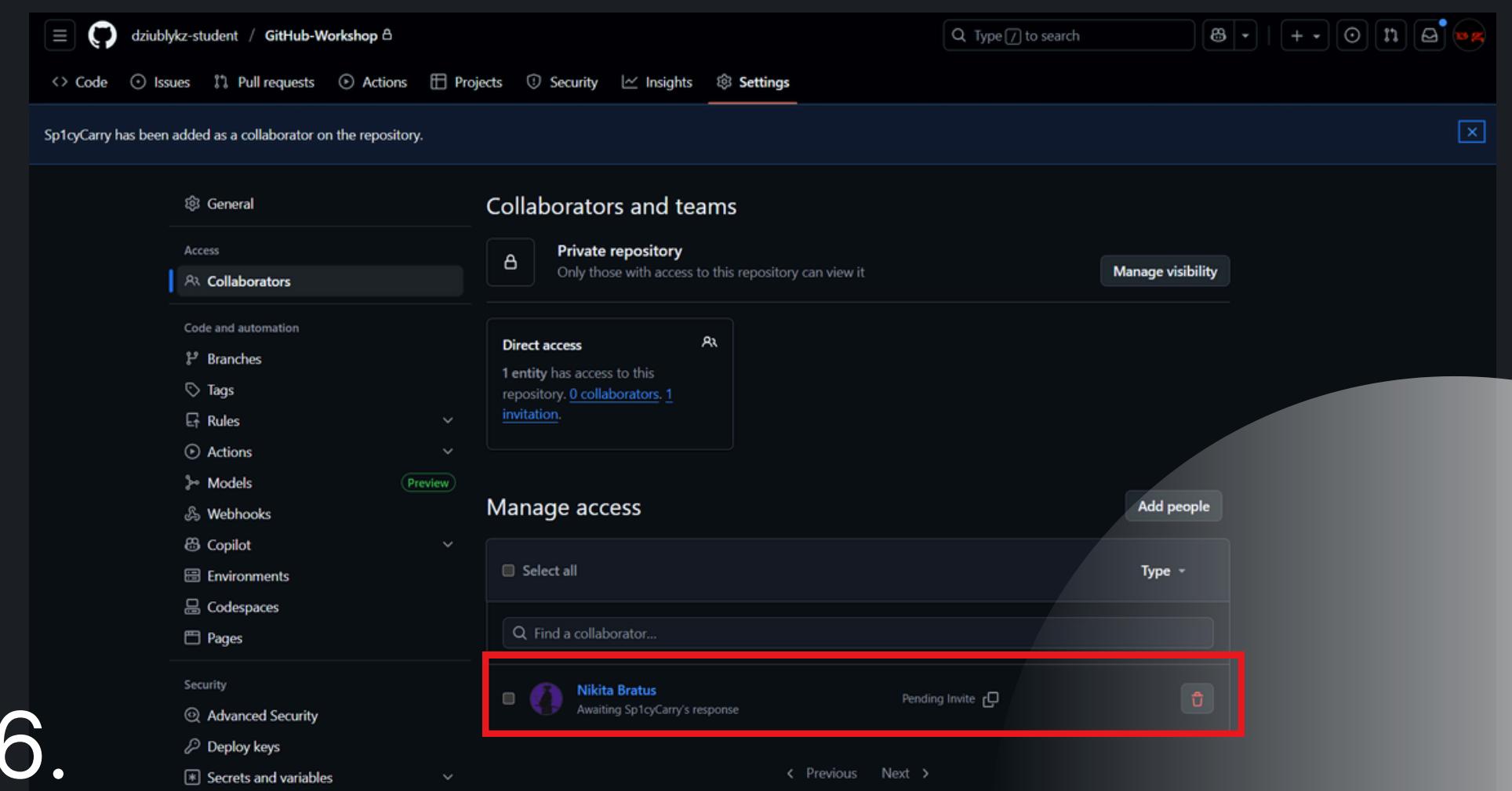
Select all

Find a collaborator...

Nikita Bratus Awaiting Sp1cyCarry's response Pending Invite

Advanced Security Deploy keys Secrets and variables

Previous Next



COLLABORATORS



COMMITTING AND PUSHING

The diagram illustrates the workflow for committing and pushing changes to a GitHub repository using GitHub Copilot in Visual Studio.

- 1. Code Changes:** In the main code editor (Program.cs), two new lines of code are added:

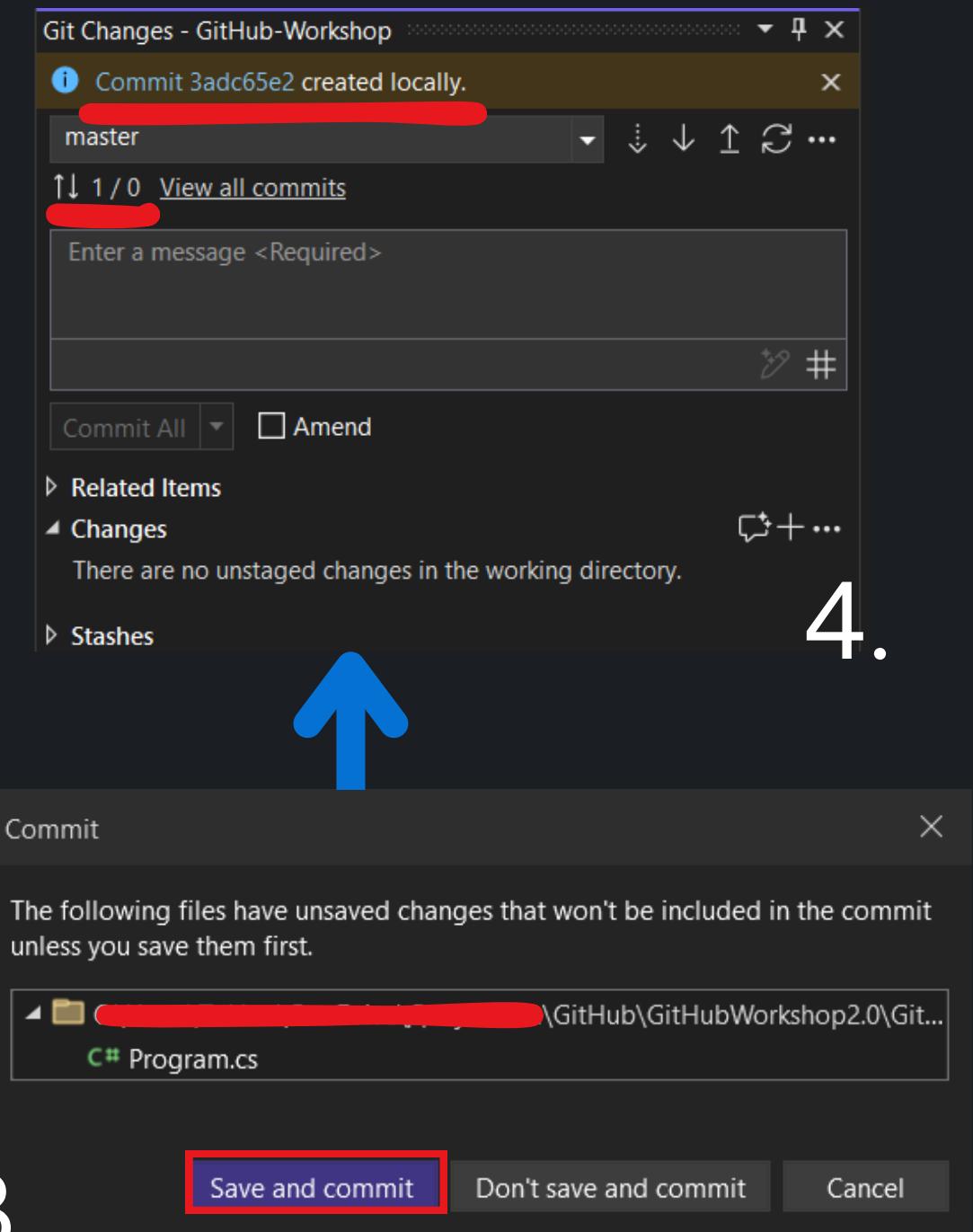
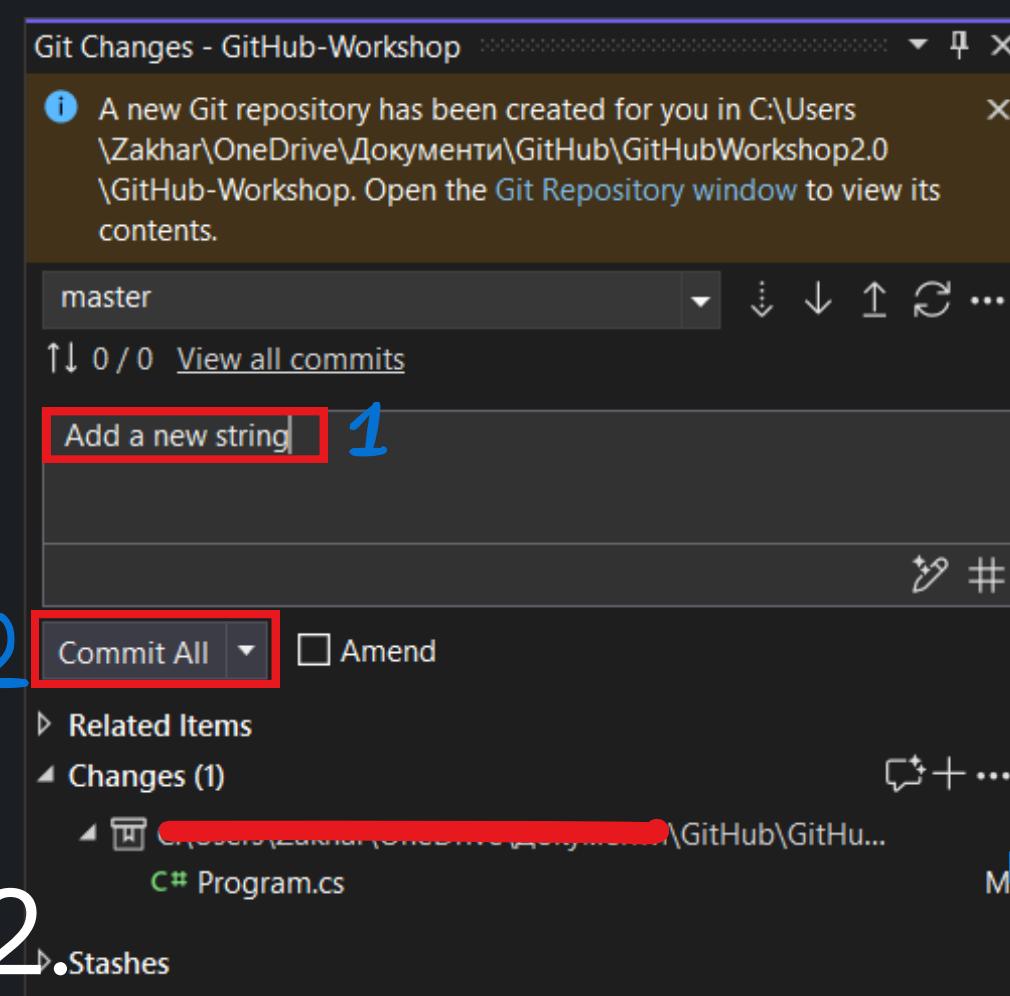
```
Console.WriteLine("Enjoy coding!");
```

A large blue arrow points from this step to the Git Changes window.
- 2. Committing:** The **Git Changes - GitHub-Workshop** window shows the changes have been detected. A red box highlights the "Changes (1)" list, which contains the new line of code.

A second blue arrow points from the Git Changes window to the Solution Explorer tab.
- 3. Solution Explorer:** The **Solution Explorer** tab shows the "Git Changes" item is selected. A third blue arrow points from the Solution Explorer tab to the **Git Changes** tab in the bottom right corner.
- 4. Git Changes Tab:** The **Git Changes** tab displays the commit message "A new Git repository has been created for you in C:\Users\Zakhar\OneDrive\Документы\GitHub\GitHubWorkshop2.0\GitHub-Workshop. Open the Git Repository window to view its contents." It also shows the "Changes (1)" list with the same new line of code, which is highlighted with a red box.



COMMITTING AND PUSHING





COMMITTING AND PUSHING

7.

GitHub-Workshop (Private)

master 1 Branch 0 Tags

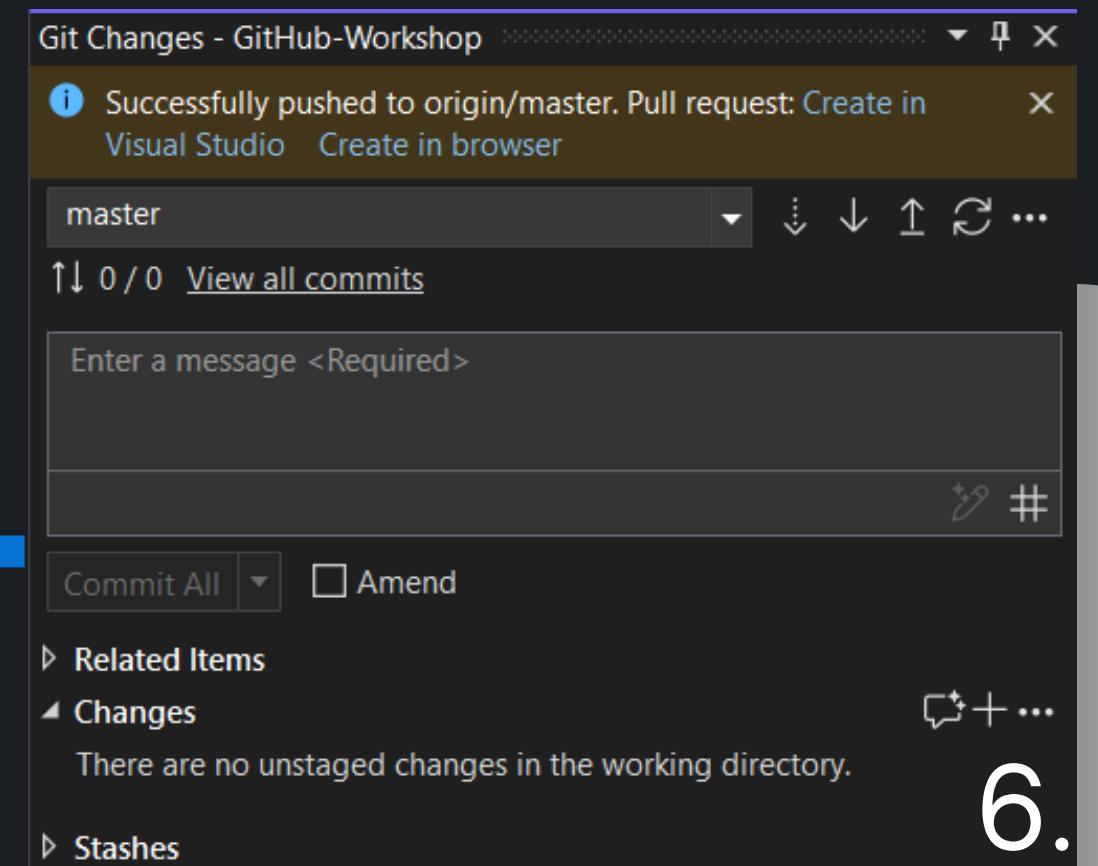
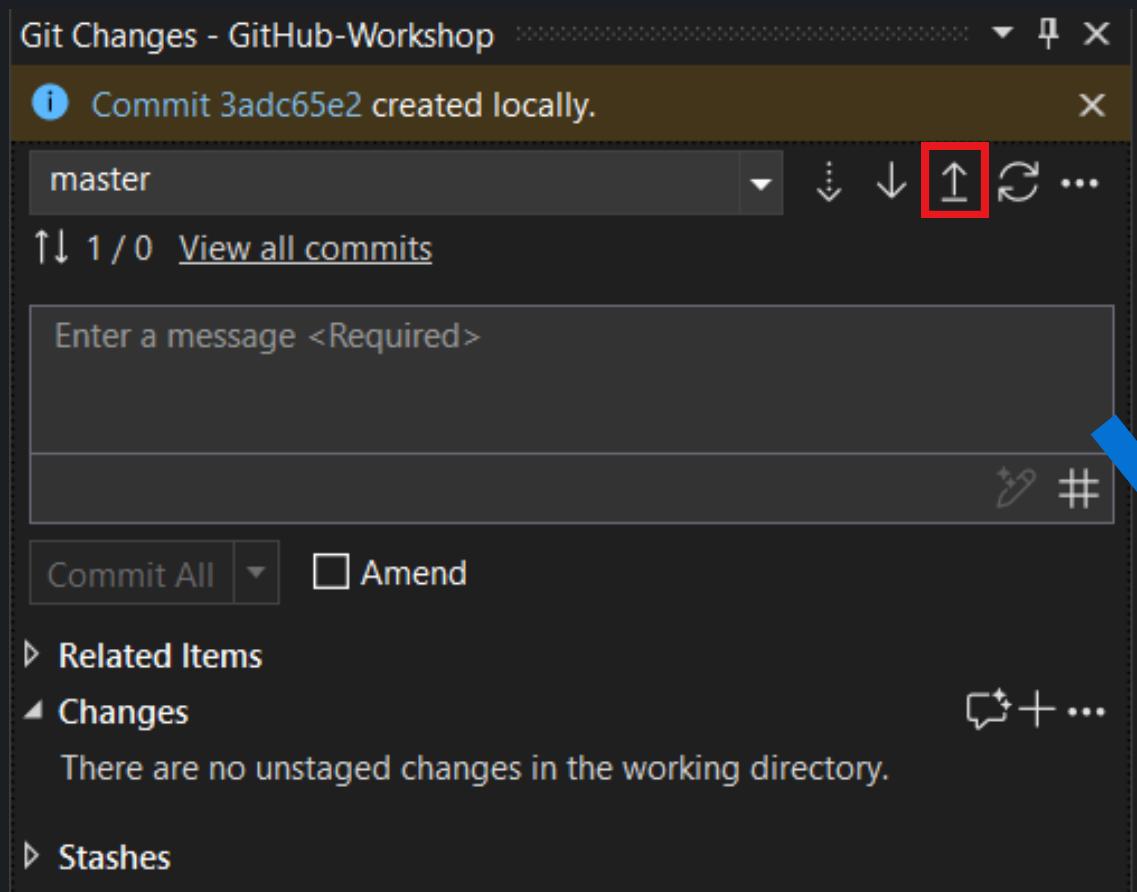
Go to file Add file Code

dziublykz add a new string ca1fac3 · now 4 Commits

- .gitattributes Add .gitattributes and .gitignore. 1 hour ago
- .gitignore Add .gitattributes and .gitignore. 1 hour ago
- GitHub-Workshop.csproj Add project files. 1 hour ago
- GitHub-Workshop.sln Add project files. 1 hour ago

Program.cs add a new string now

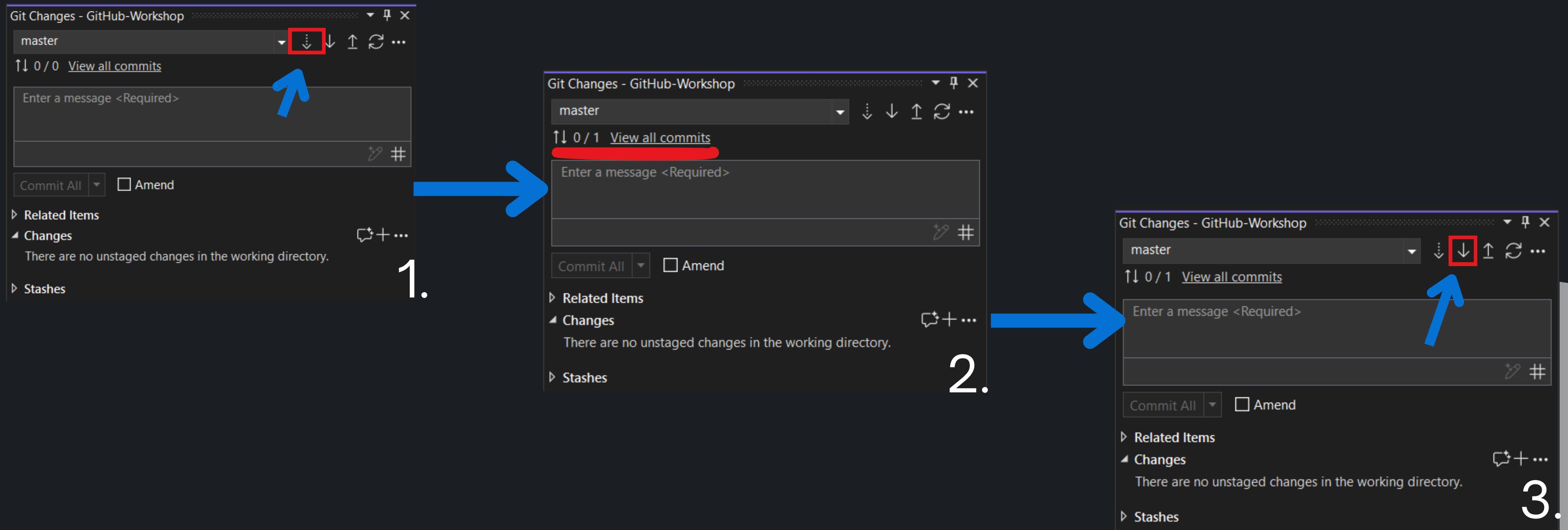
5.



6.



PULLING CHANGES





PULLING CHANGES

The screenshot shows the Visual Studio IDE interface with the following details:

- File Bar:** File, Edit, View, Git, Project, Build, Debug, Test, Analyze, Tools, Extensions, Window, Help.
- Search Bar:** Search.
- Toolbar:** Includes icons for Undo, Redo, Save, and others.
- Project Explorer:** Shows the project "GitHub-Workshop" with the file "Program.cs" open.
- Code Editor:** Displays the following C# code in "Program.cs":

```
1  namespace GitHub_Workshop
2  {
3      internal class Program
4      {
5          static void Main(string[] args)
6          {
7              Program program = new Program();
8              program.Start();
9          }
10         void Start()
11         {
12             Console.WriteLine("Hello, World!");
13             Console.WriteLine("Enjoy coding!");
14
15             Console.WriteLine("Hello, GitHub Workshop!");
16         }
17     }
18 }
19 }
```

The code editor has several lines highlighted with red bars, indicating changes or differences.

Git Changes - GitHub-Workshop panel:

- Repository updated to commit 8dc46f3d.
- master
- 0 / 0 View all commits
- Enter a message <Required>
- Commit All ▾ Amend
- Related Items
- Changes: There are no unstaged changes in the working directory.
- Stashes

A large number "4." is visible in the bottom right corner of the screenshot.



GITHUB DESKTOP

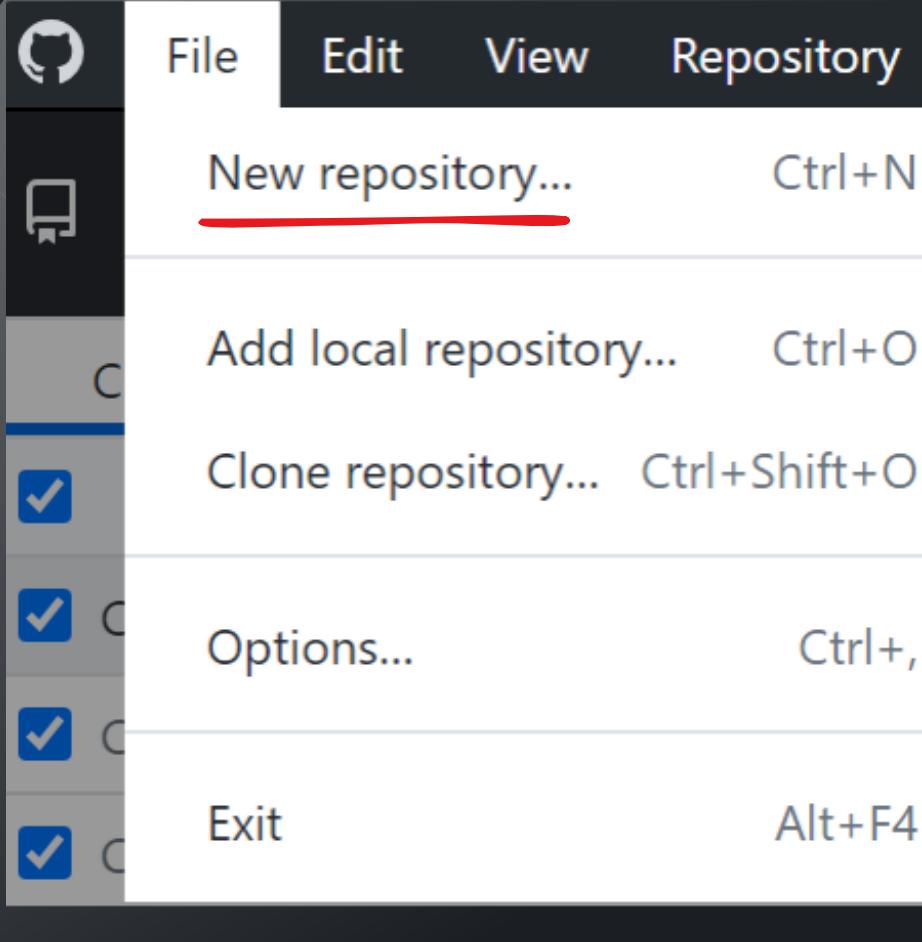
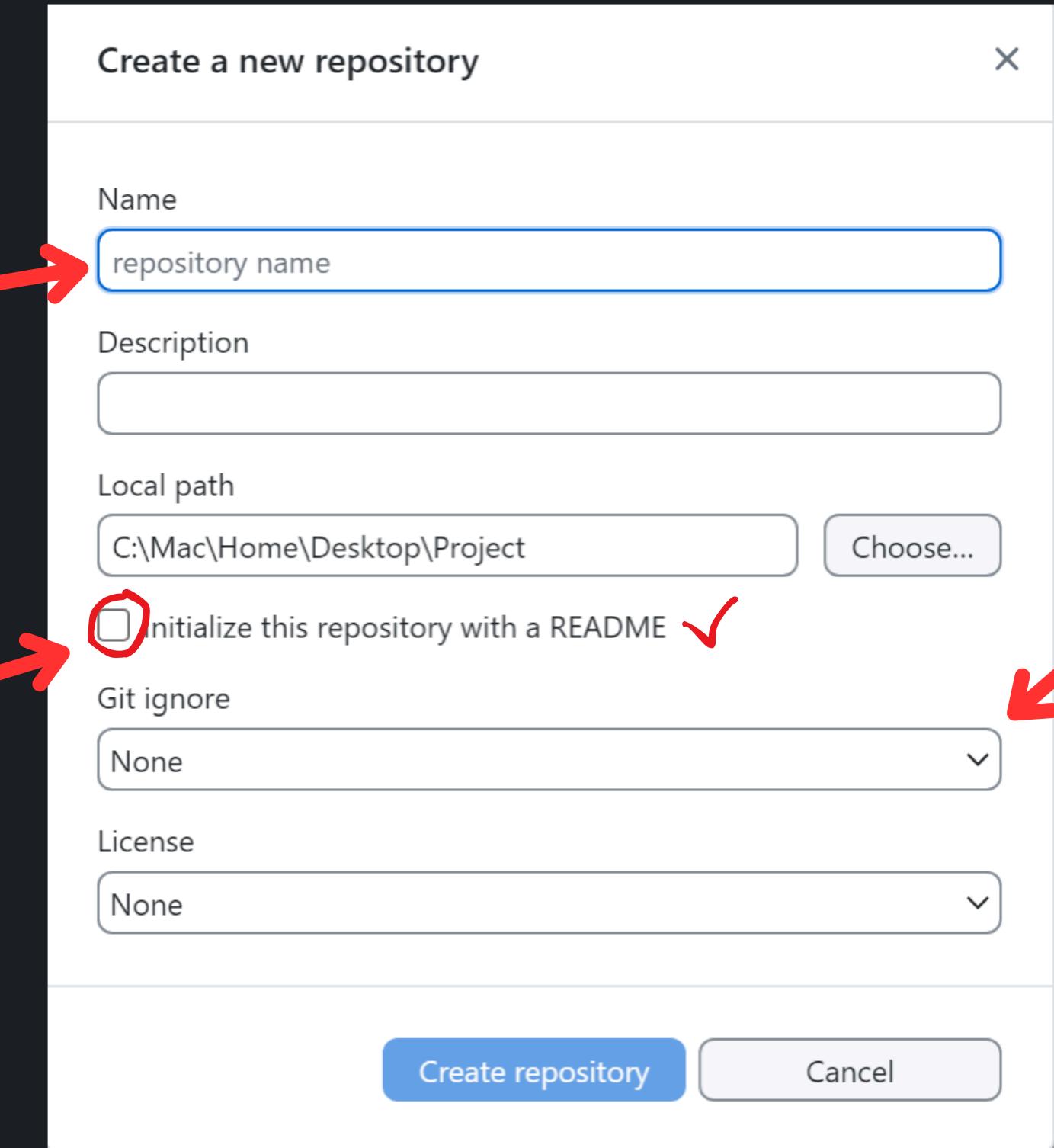
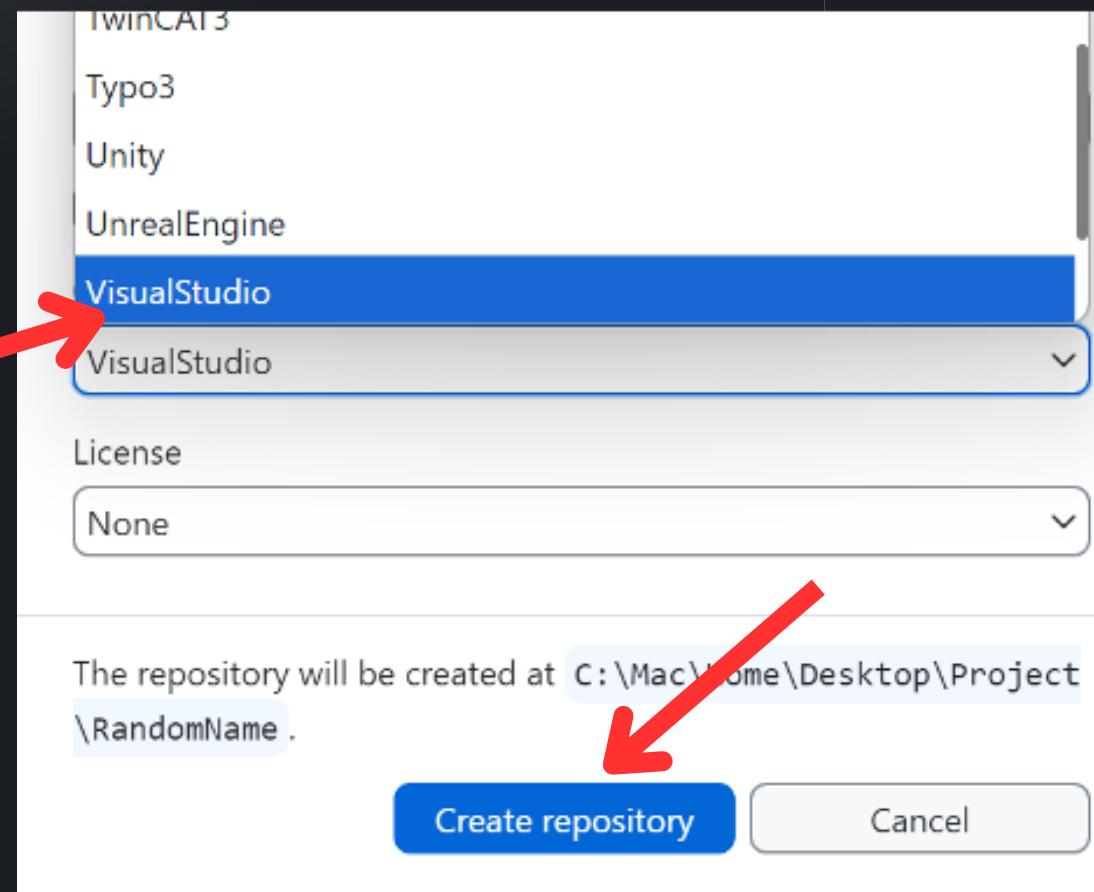
- Open GitHub Desktop, click “File → New Repository.”
- Add a name, description, and create a .gitignore or license file.
- Repository appears locally.





CREATING A REPOSITORY IN GITHUB OPTION 1

Page 23

1. 
2. 
3. 

The repository will be created at `C:\Mac\Home\Desktop\Project\RandomName`.

Create repository **Cancel**



.GITIGNORE

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

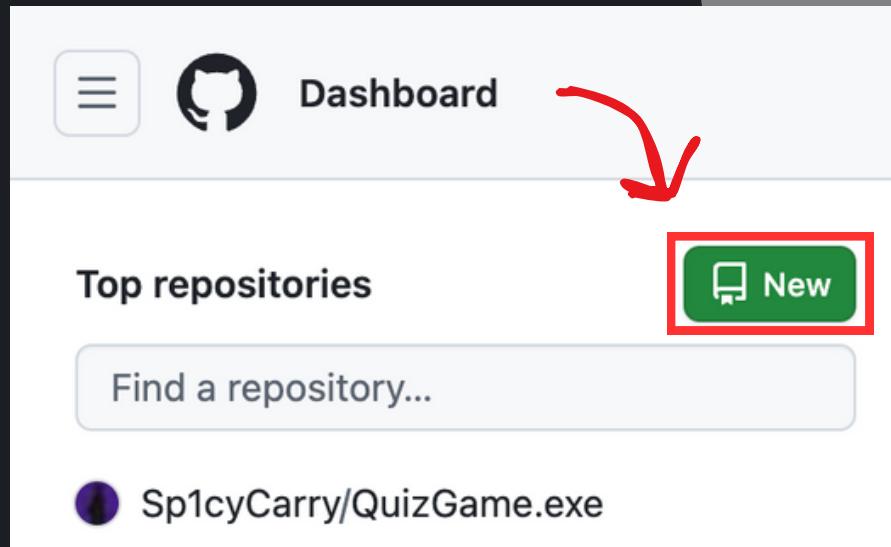
- Used for build files, logs, secrets, temp files
- Keeps the repository clean
- Can be edited at any time





CREATING A REPOSITORY IN GITHUB OPTION 2

1.



2.

Create a new repository

Repositories contain a project's files and version history. Have a project elsewhere? [Import a repository](#).
Required fields are marked with an asterisk (*).

1 General

Owner * Sp1cyCarry Repository name * 1. MyNewRepo is available.

Great repository names are short and memorable. How about [laughing-octo-doodle](#)?

Description

0 / 350 characters

2 Configuration

Choose visibility * 2. Private

Choose who can see and commit to this repository

Add README 3. On

READMEs can be used as longer descriptions. [About READMEs](#)

Add .gitignore 4. VisualStudio

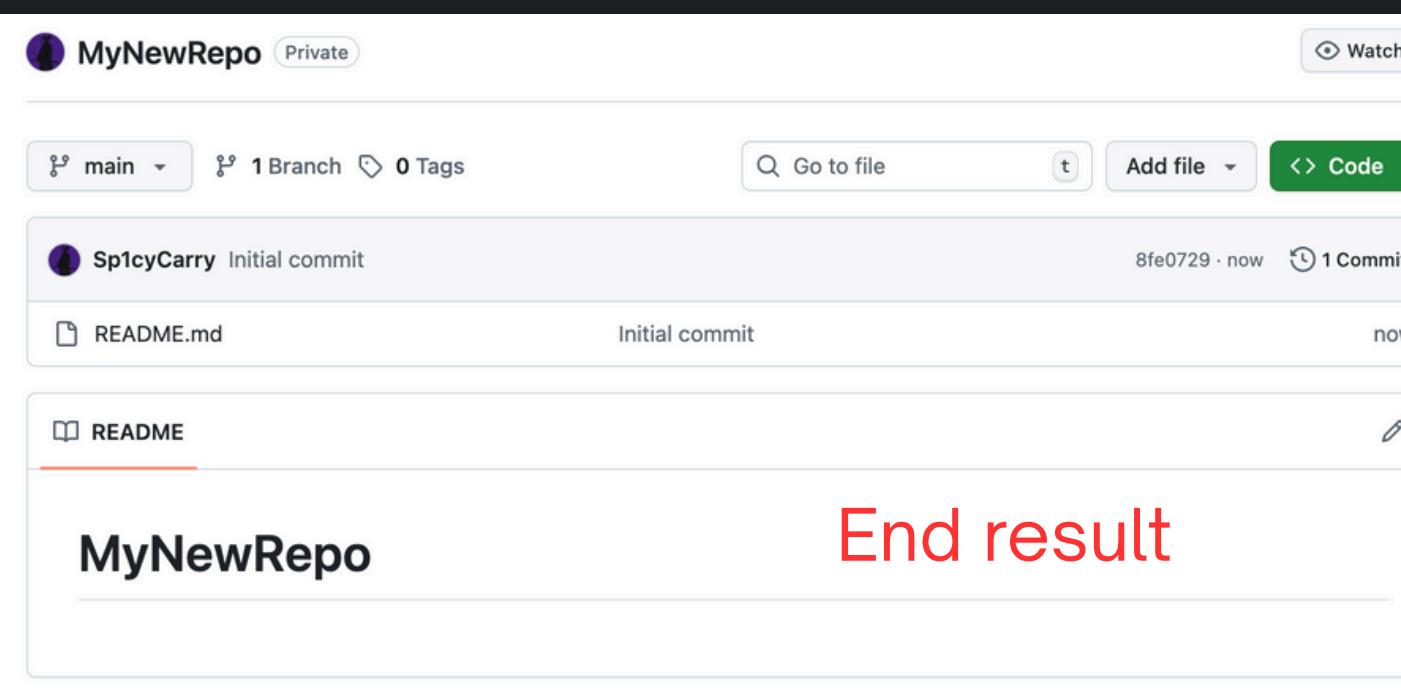
.gitignore tells git which files not to track. [About ignoring files](#)

Add license 5. No license

Licenses explain how others can use your code. [About licenses](#)

Create repository

3.

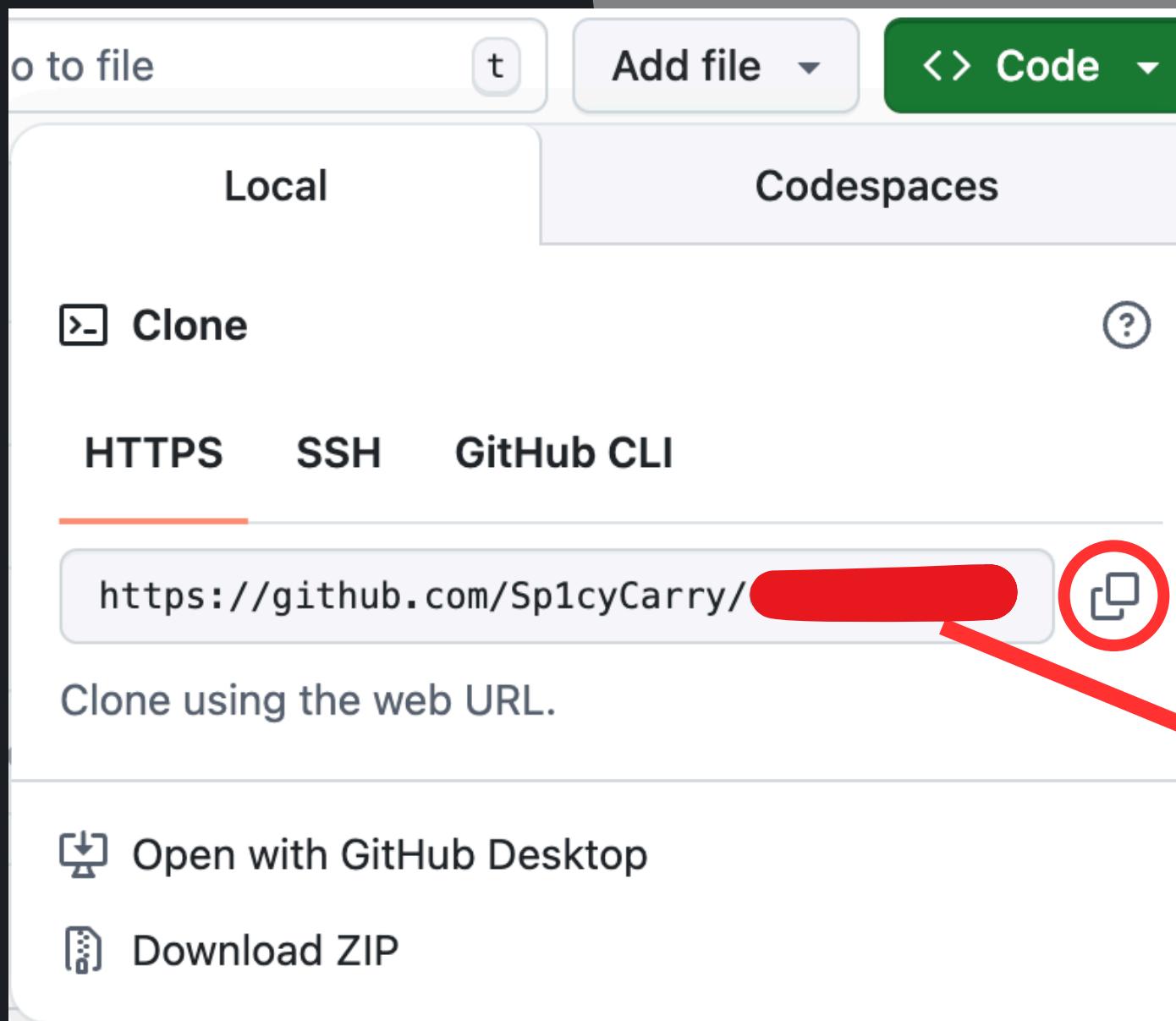


End result

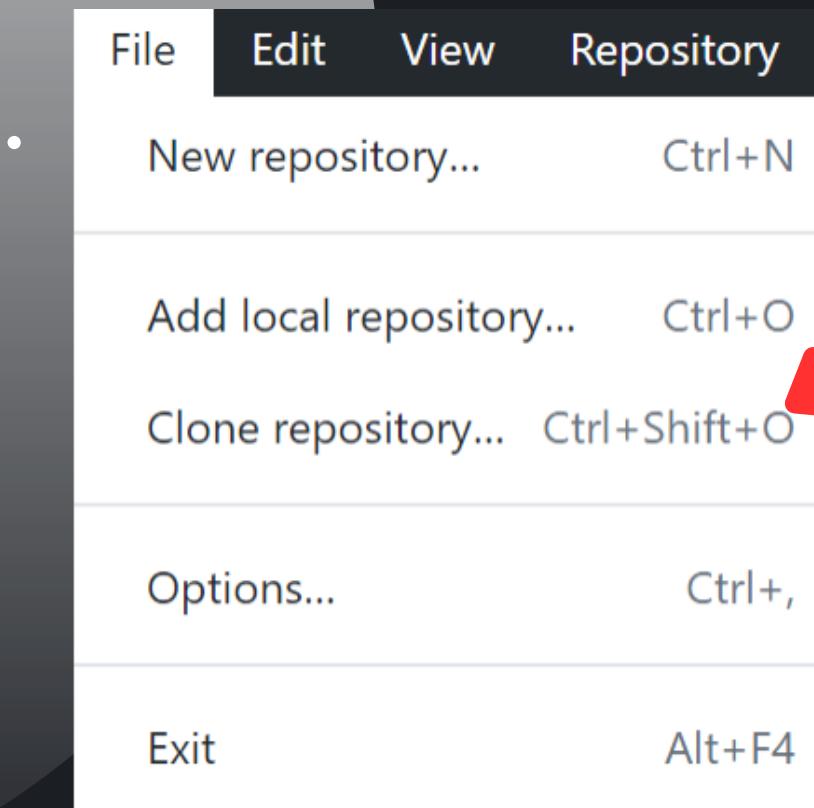


CREATING A REPOSITORY IN GITHUB OPTION 2

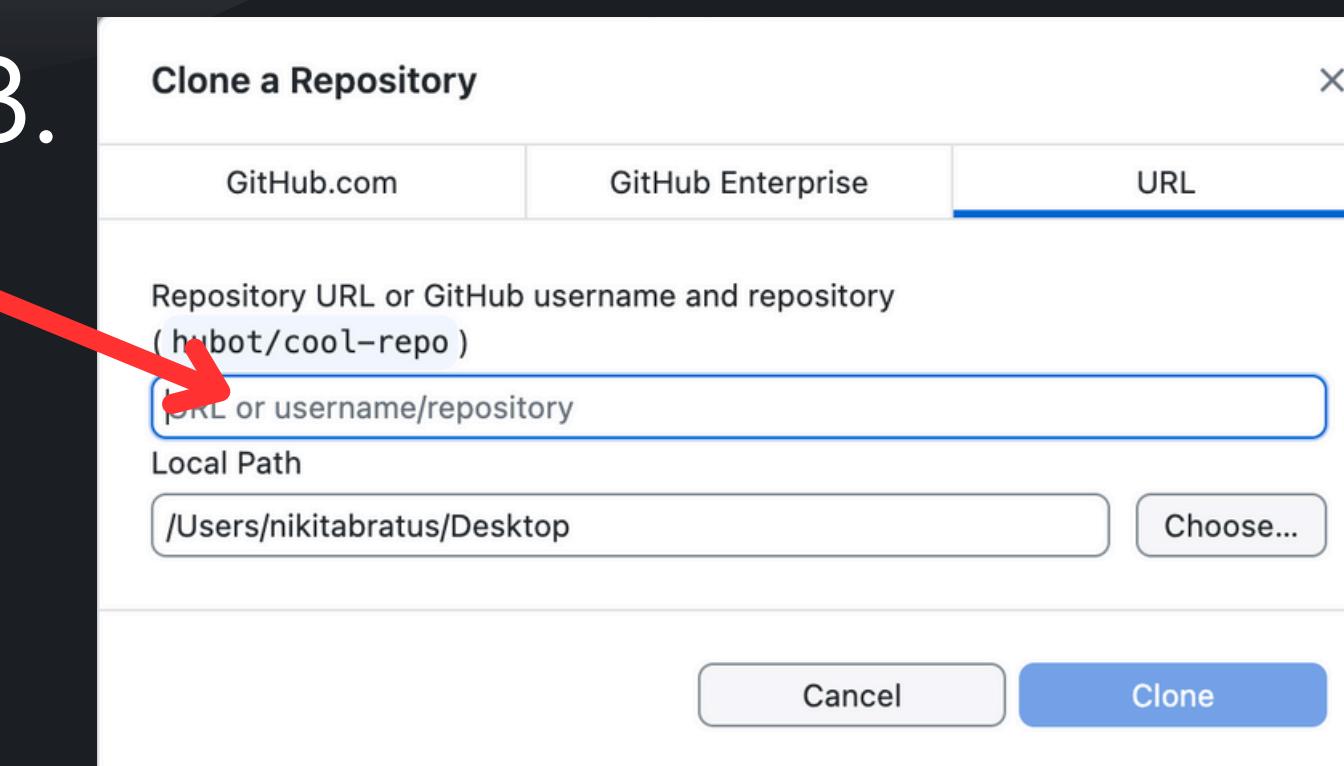
1.



2.



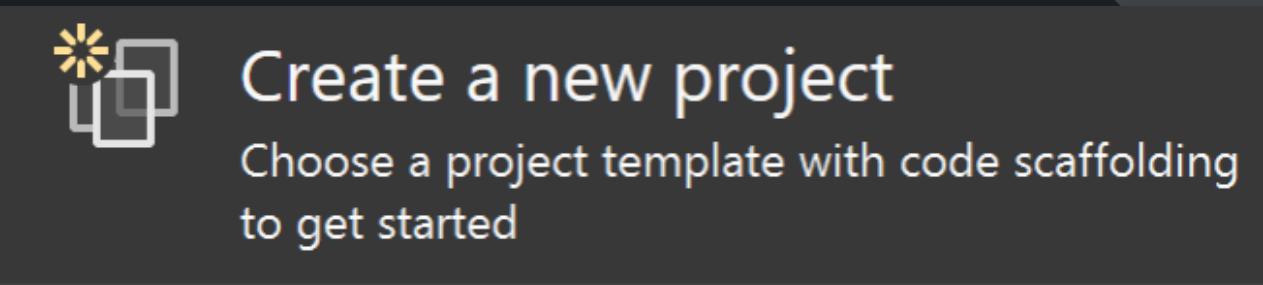
3.





CONNECTING EMPTY REPO TO THE SOLUTION

Adding files where we can change something:

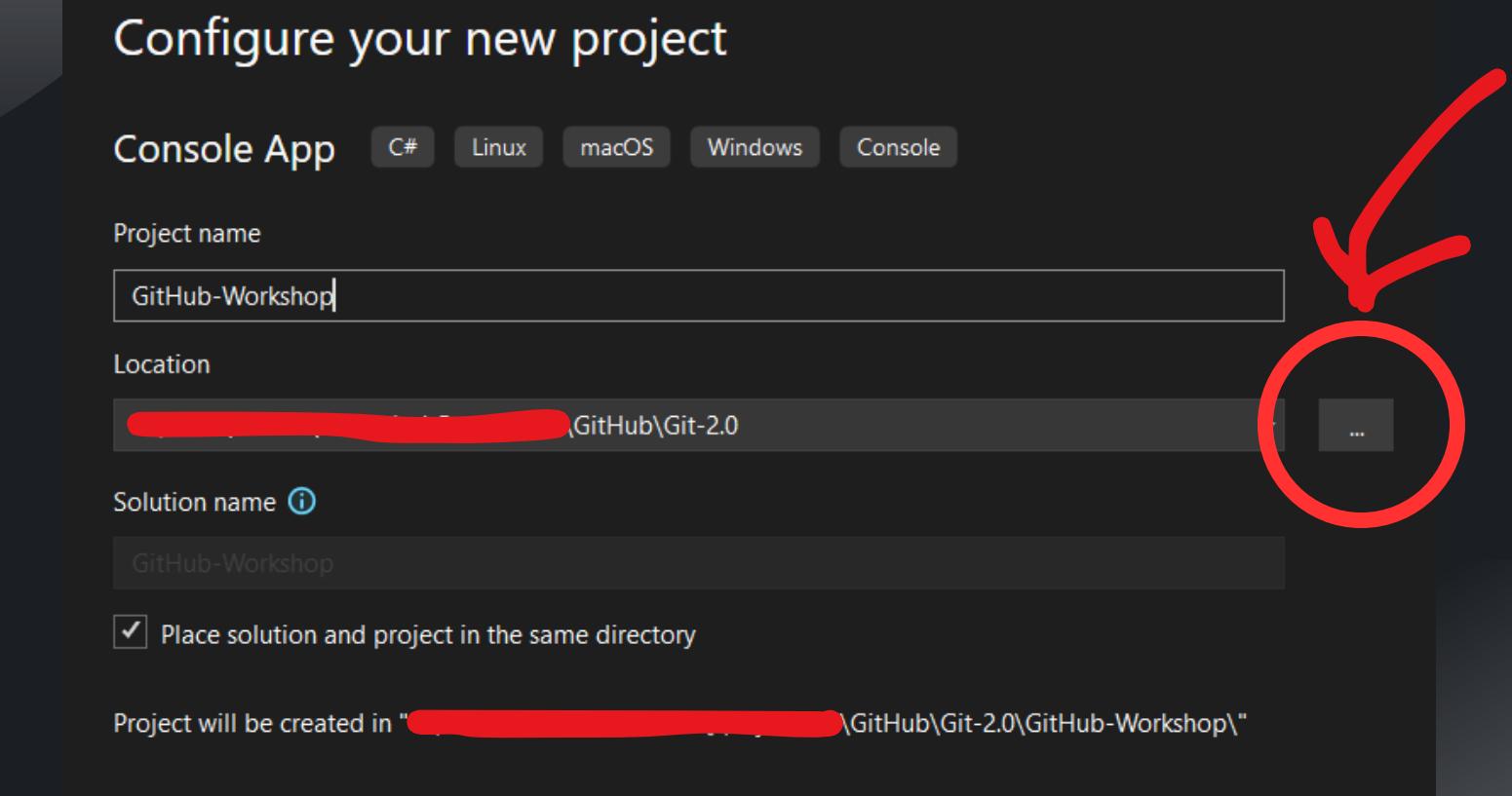


Make just a small change:

```
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

No local changes

There are no uncommitted changes in this repository. Here are some friendly suggestions for what to do next.

Open the repository in your external editor
Select your editor in [Settings](#)

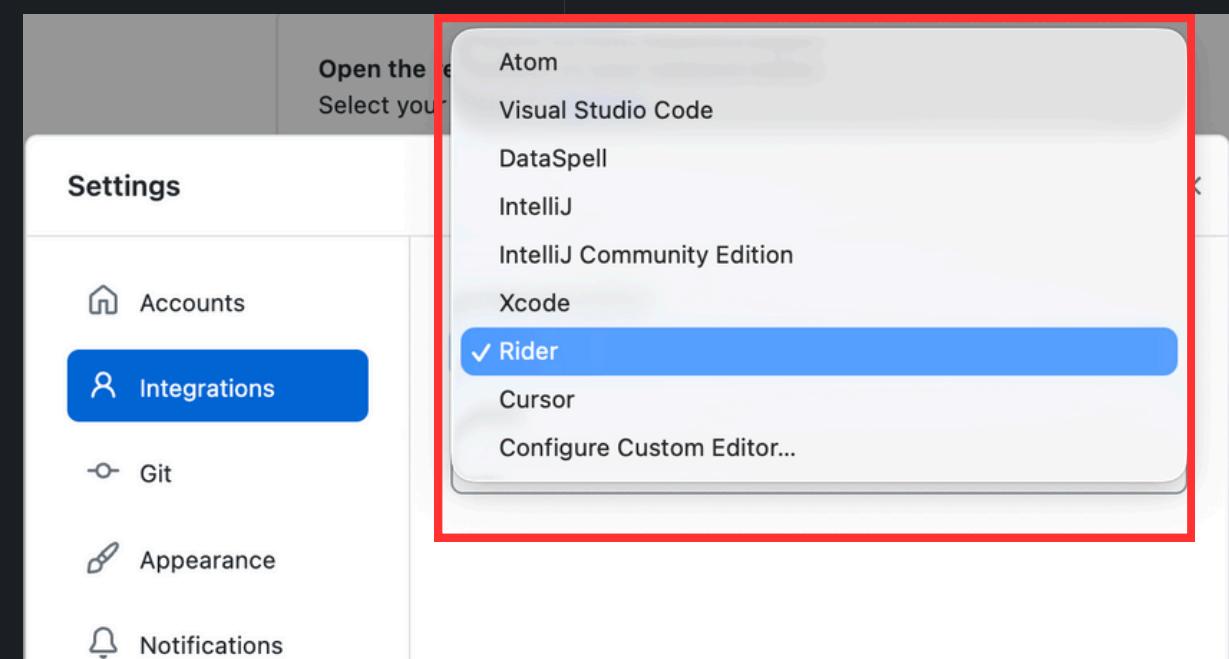
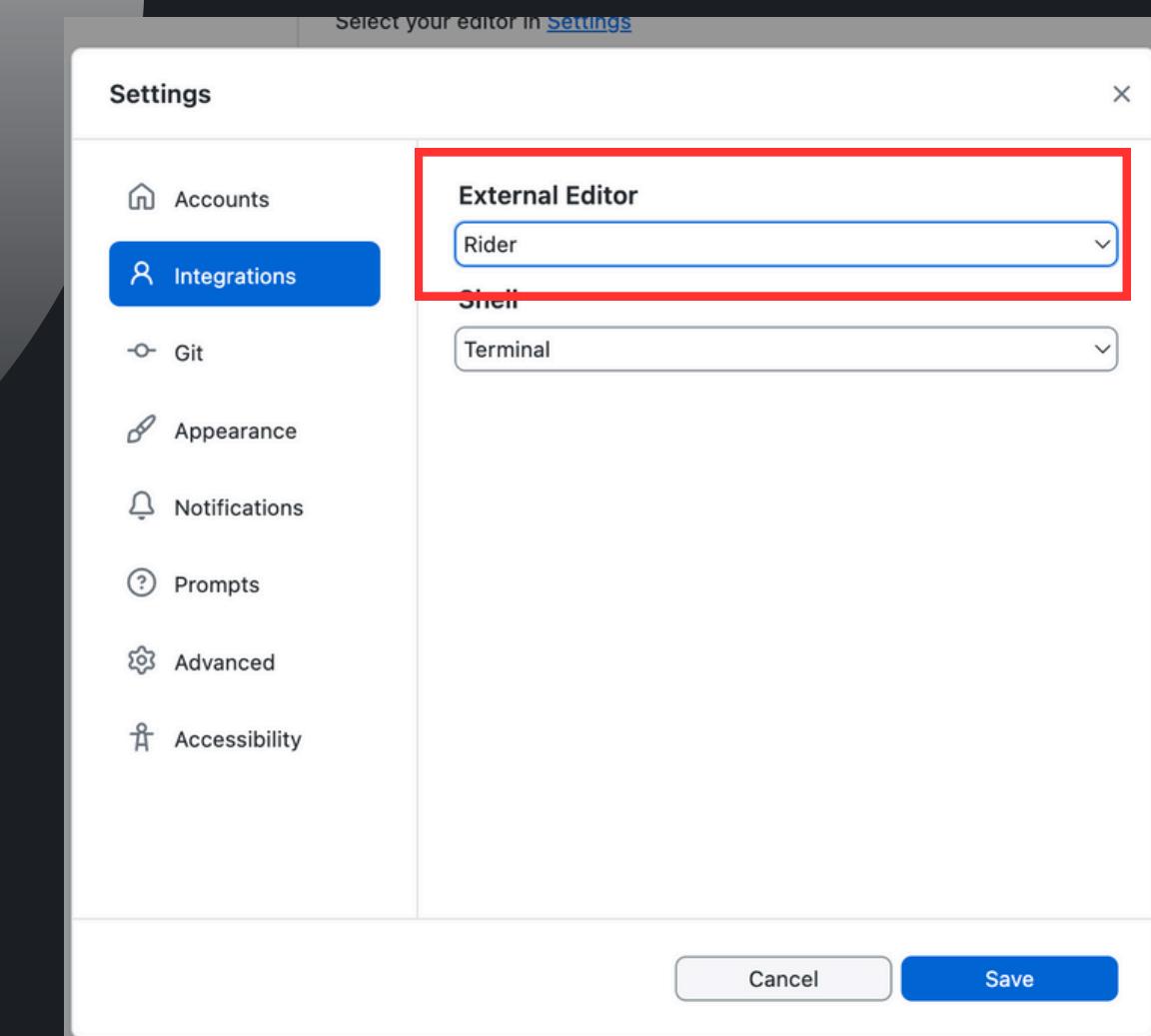
Repository menu or ⌘ ⌘ A

View the files of your repository in Finder
Repository menu or ⌘ ⌘ F

Show in Finder

Open the repository page on GitHub in your browser
Repository menu or ⌘ ⌘ G

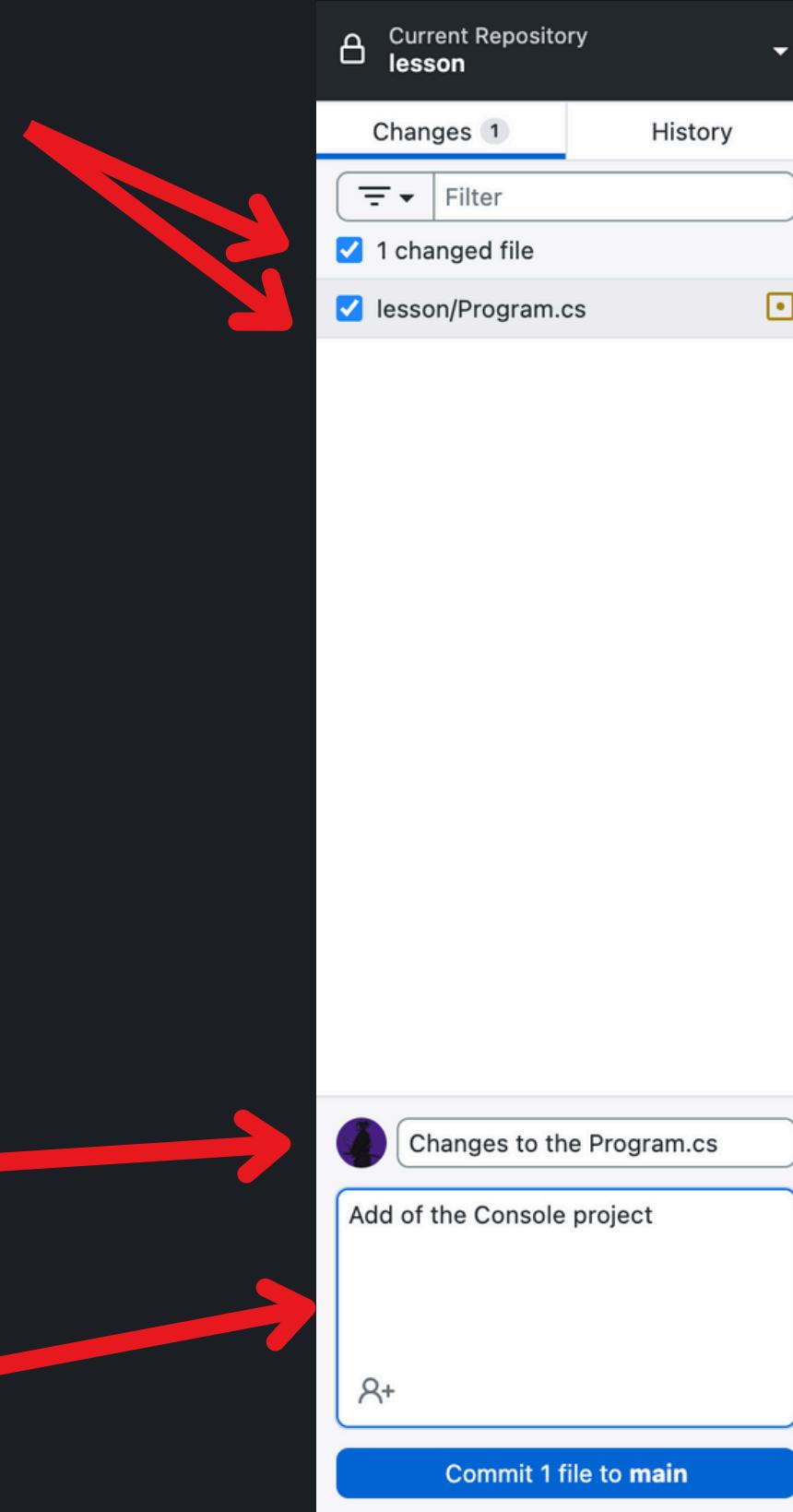
View on GitHub





COMMITTING CHANGES AND PUSHING TO GITHUB

Select the files which you want
to add to commit



Add a Commit header



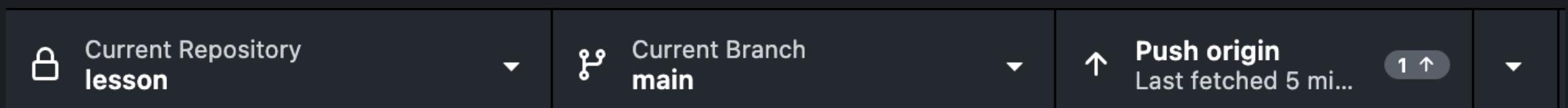
Add a Commit description





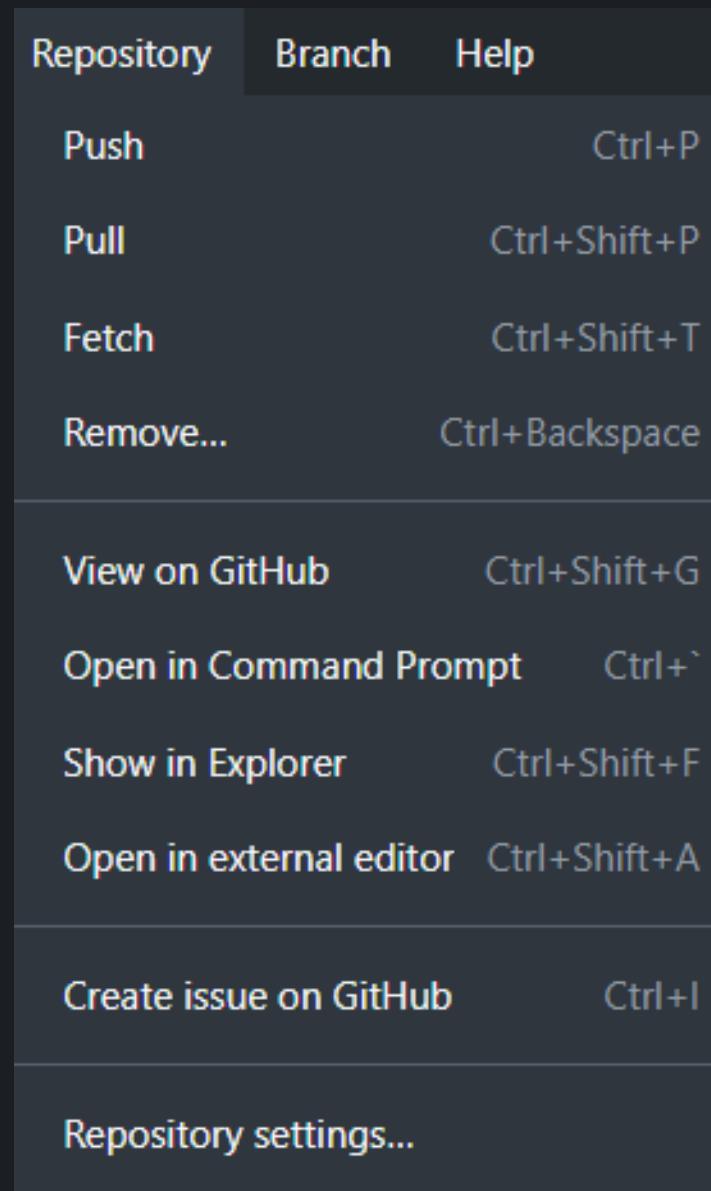
COMMITTING CHANGES AND PUSHING TO GITHUB

Push the changes





PULLING THE CHANGES

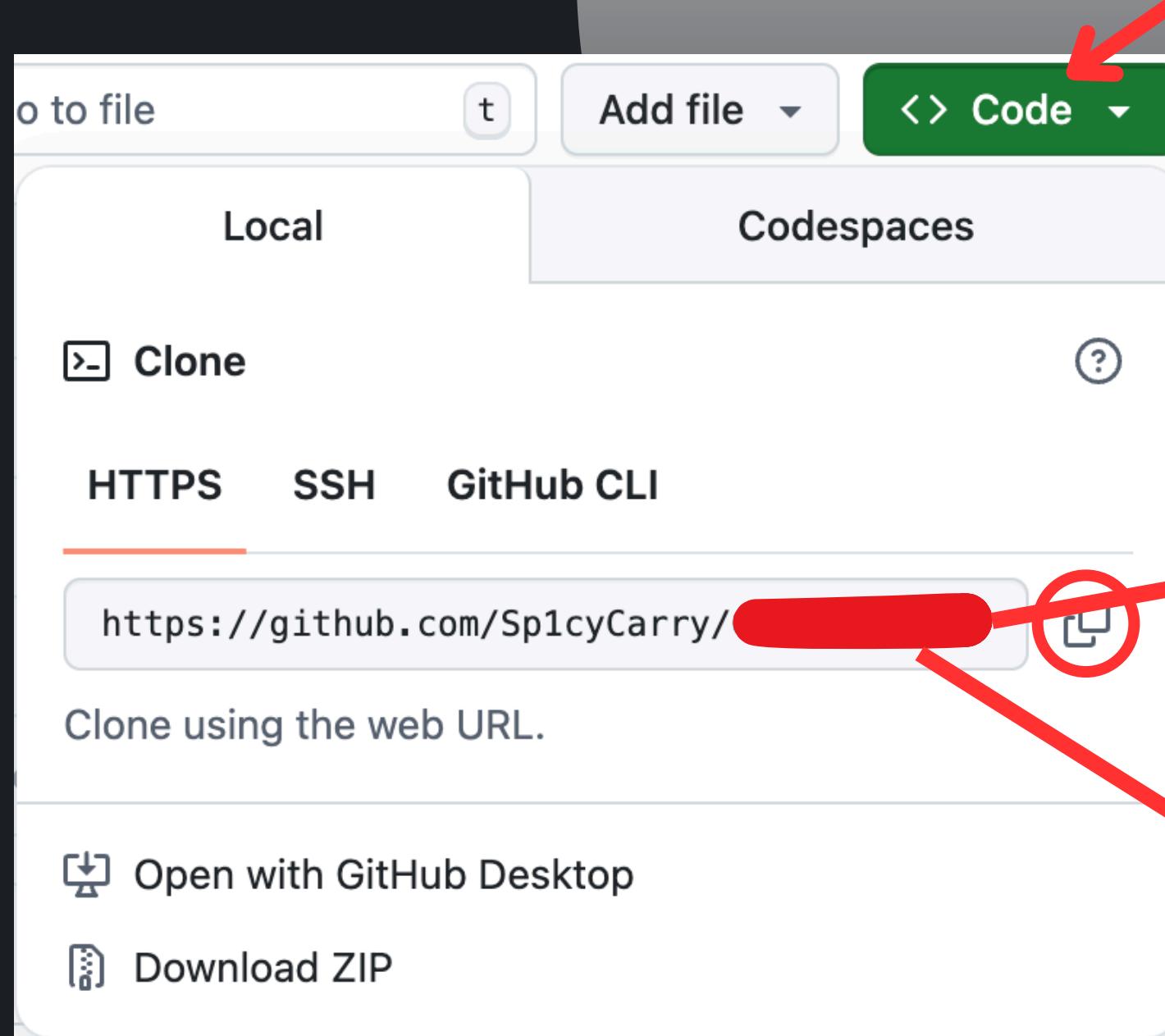


To pull the changes click on the pull button located in the Repository section.

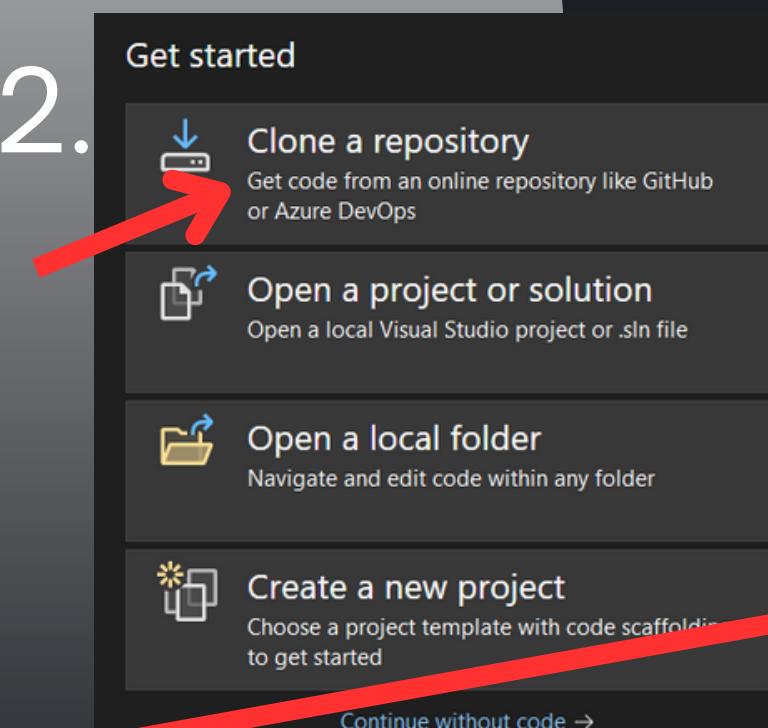


CLONING REPO IN VISUAL STUDIO 2022

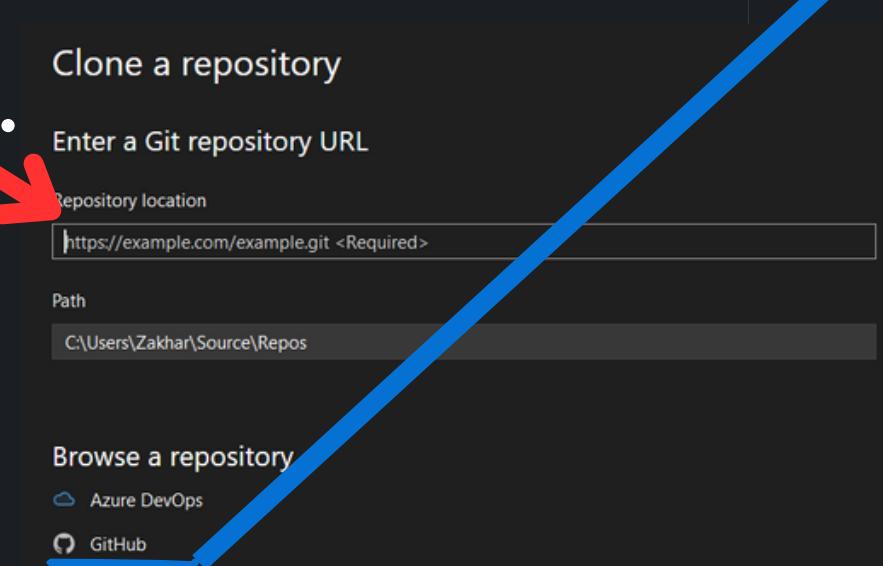
1.



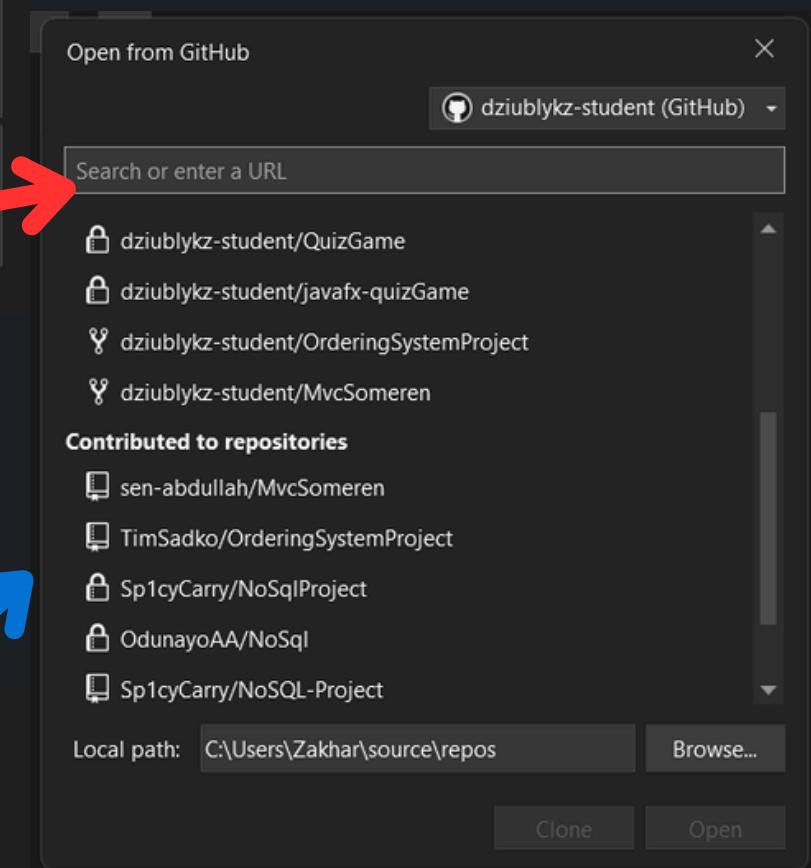
2.



3.



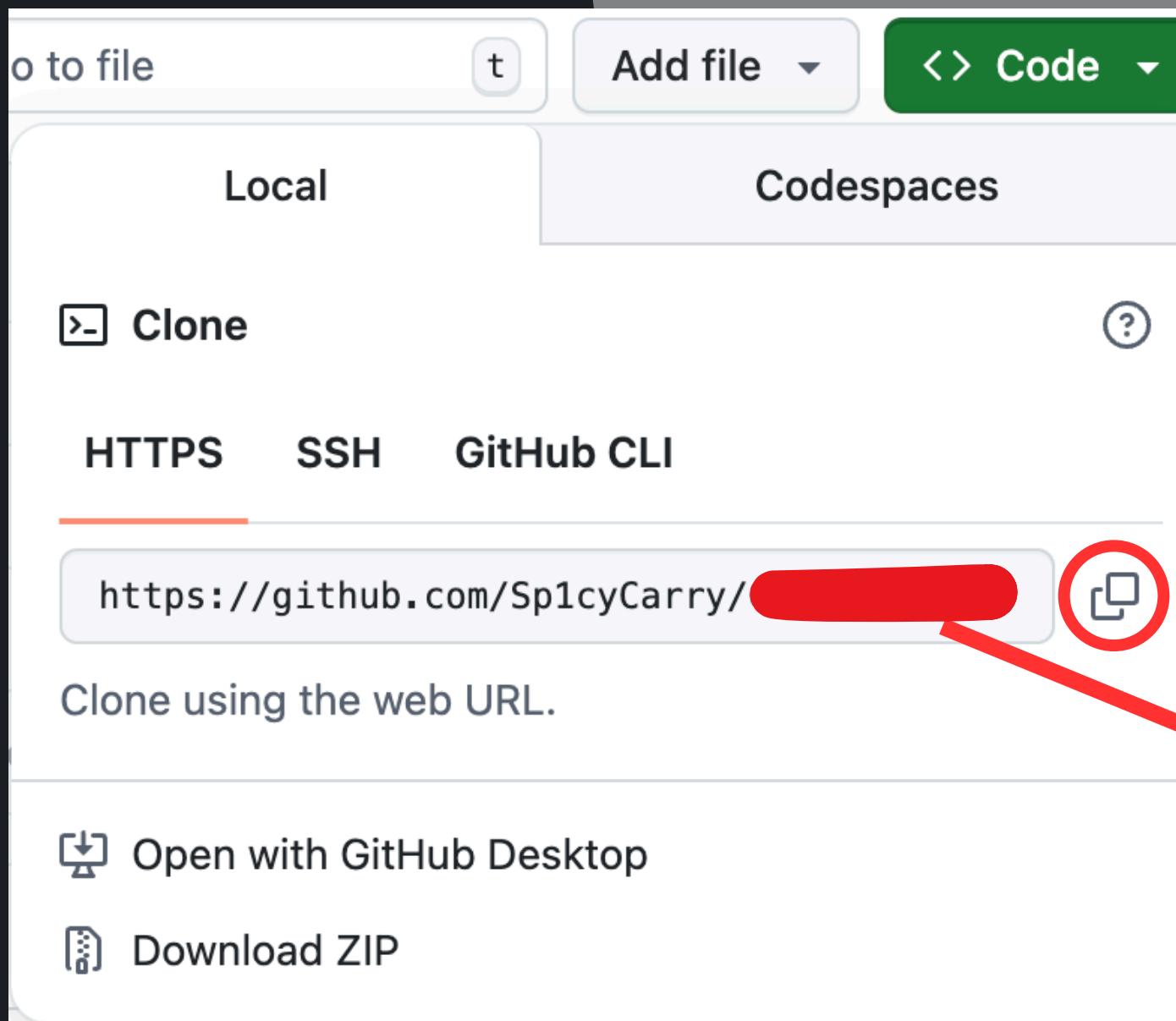
4.



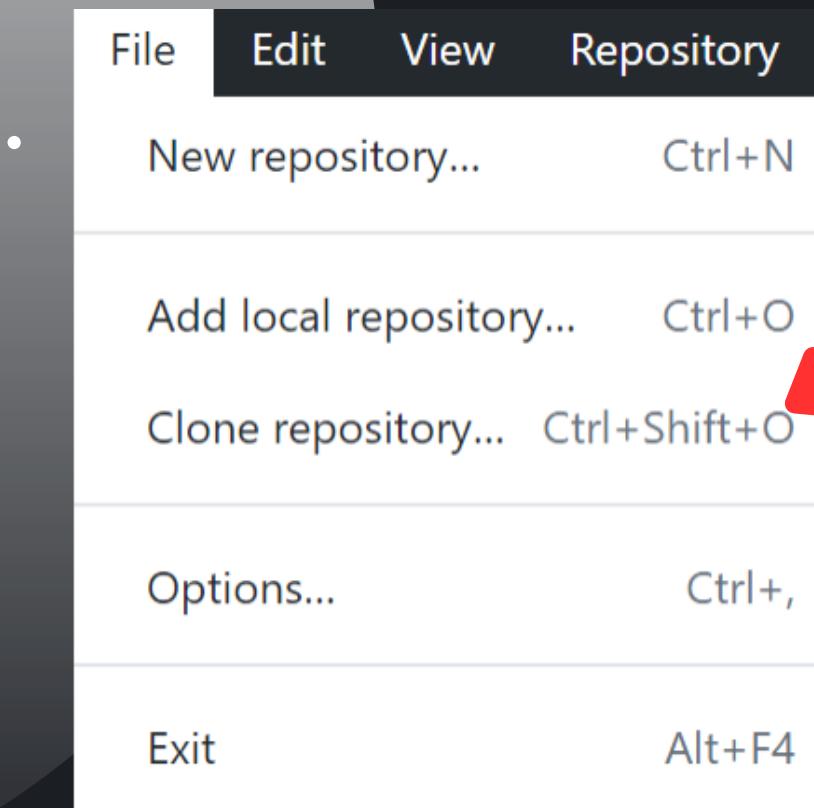


CLONING REPO WITH GITHUB DESKTOP

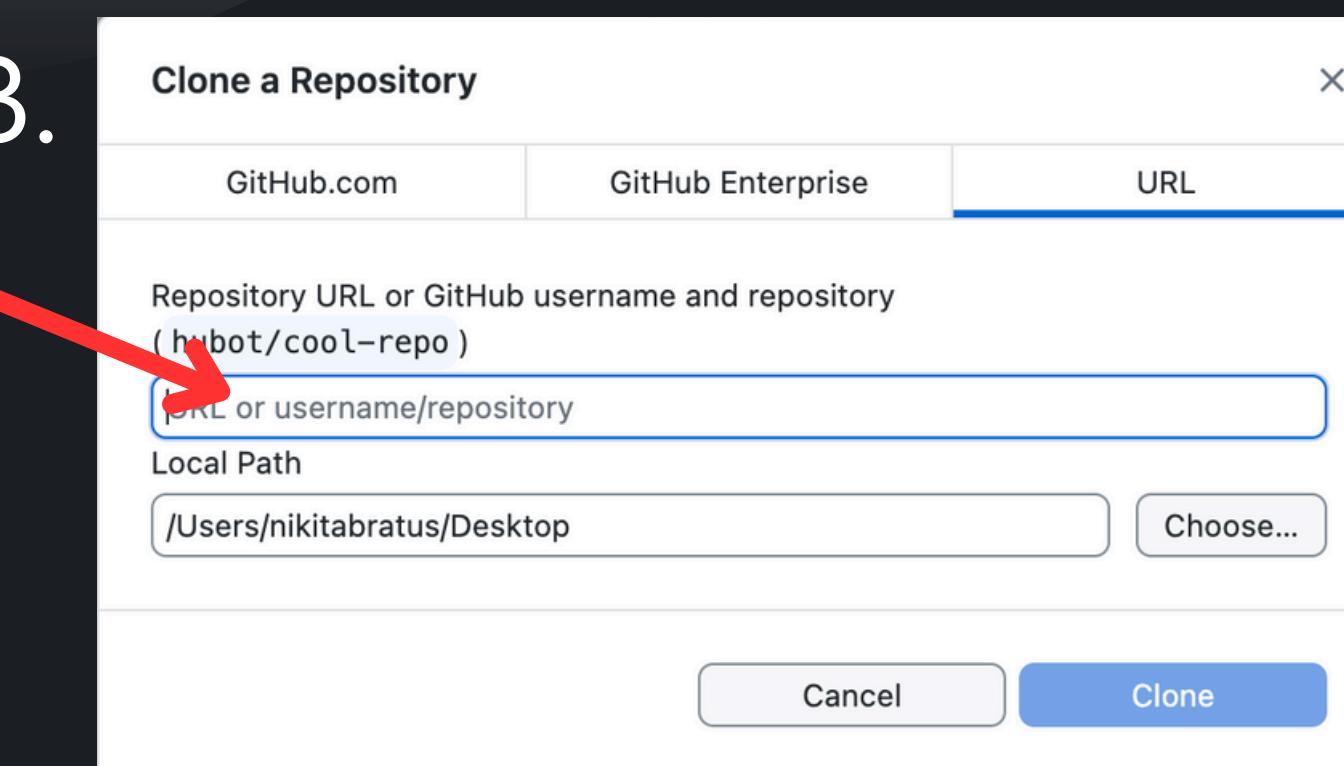
1.



2.



3.





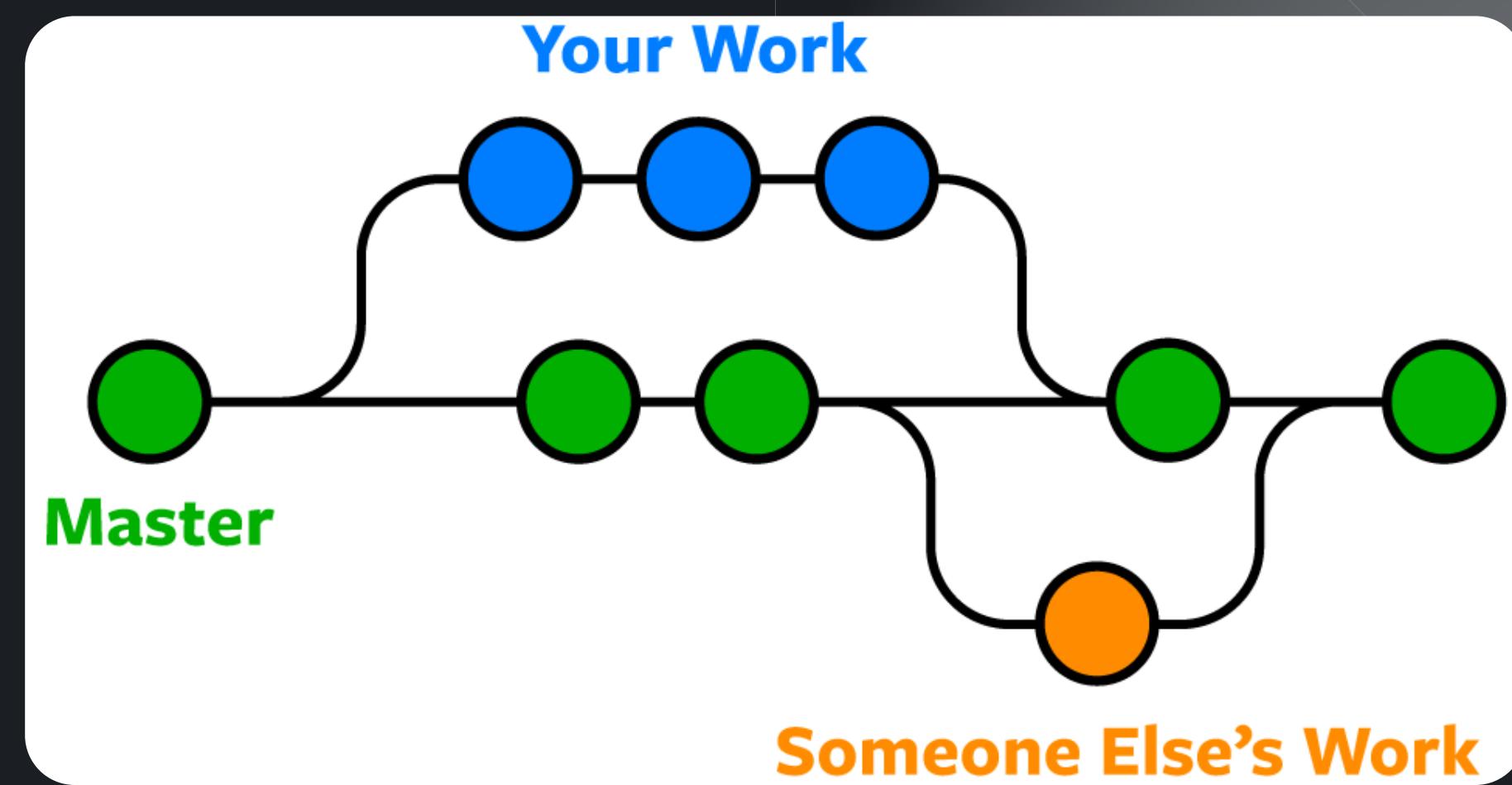
WHAT ARE BRANCHES?



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.

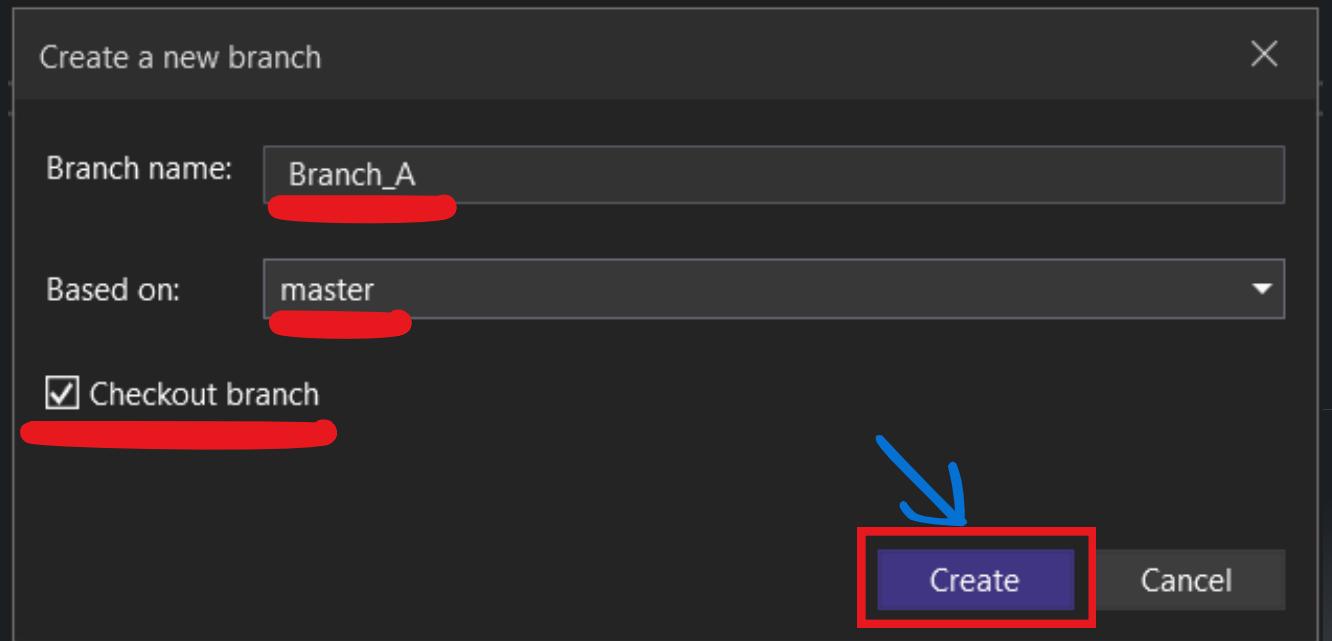
Branches allow experimentation and prevent accidental damage to the project's primary version.





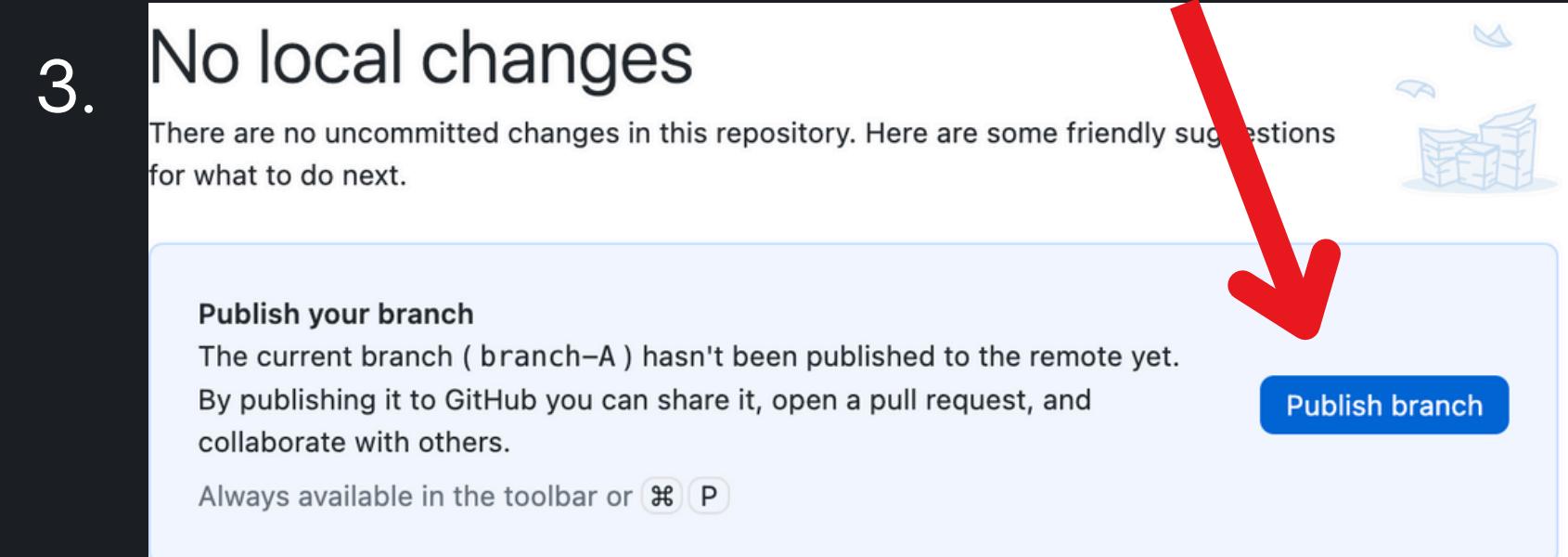
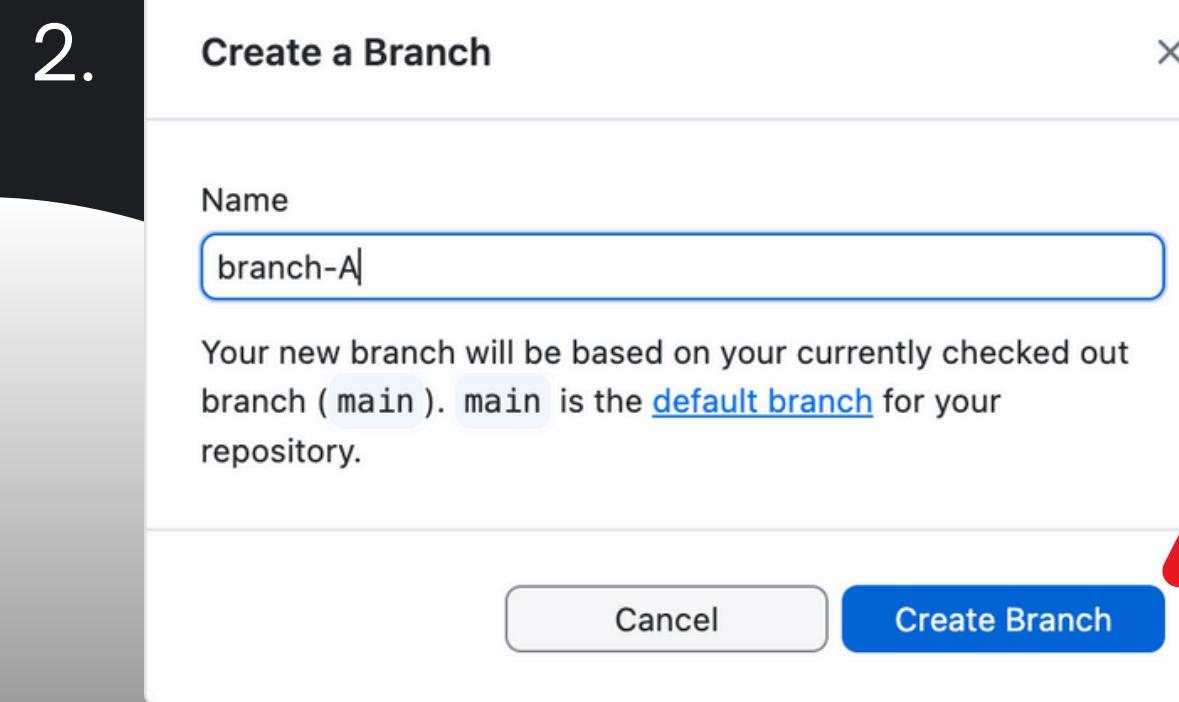
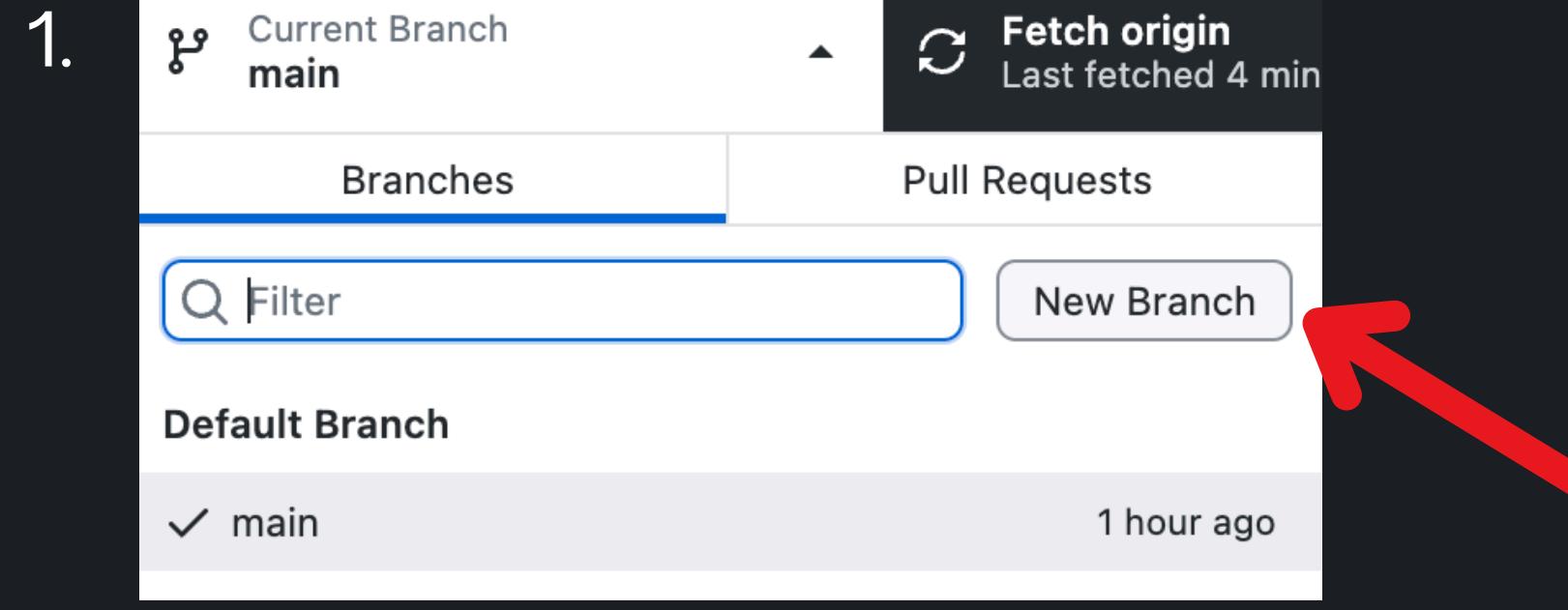
CREATING BRANCHES IN VISUAL STUDIO

A screenshot of the Visual Studio IDE. The top menu bar includes File, Edit, View, Git, Project, Build, Debug, Test, Analyze, Tools, Extensions, Window, Help, and a Search bar. The main window shows a code editor with 'Program.cs' containing C# code. The status bar at the bottom indicates 'master'. On the right side, there's a 'Git Changes' sidebar with a 'New Branch' button highlighted by a red box. A large blue arrow points from this button to a separate dialog window.





CREATING BRANCHES IN GITHUB DESKTOP

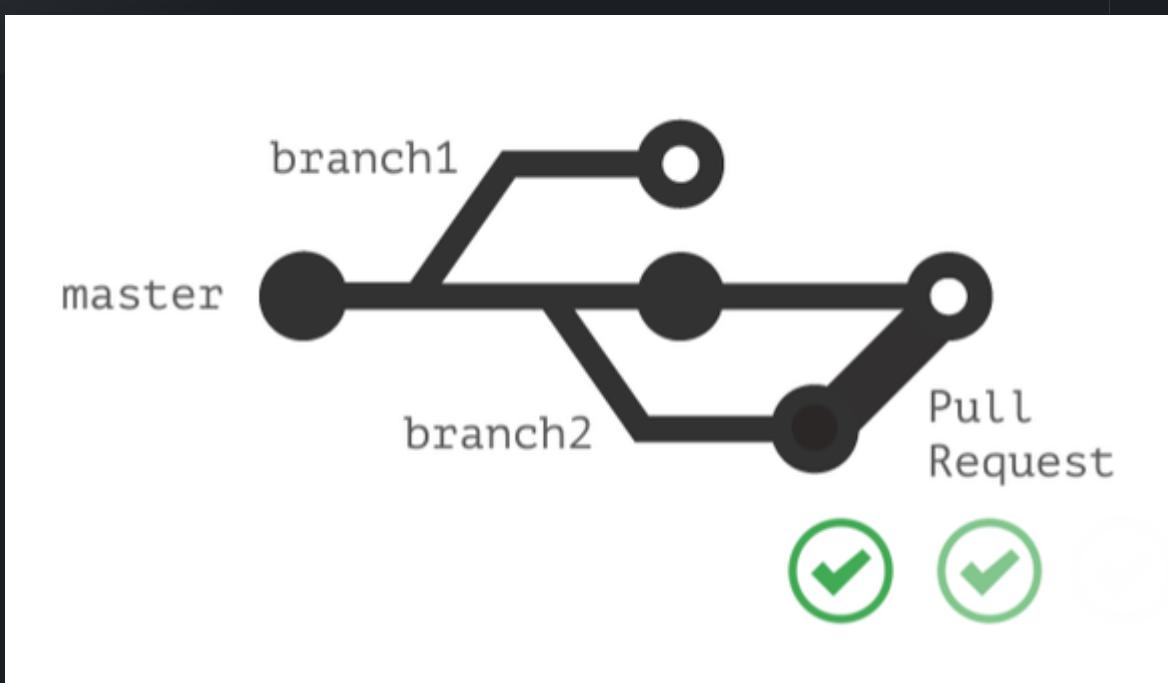




WHAT IS A PULL REQUEST?

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

Create a Pull Request from your current branch
The current branch (branch-A) is already published to GitHub. Create a pull request to propose and collaborate on your changes.

Branch menu or ⌘ R

Create Pull Request

Made a small exit button #2

Open Sp1cyCarry wants to merge 1 commit into main from branch-A

Conversation 0 Commits 1 Checks 0 Files changed 2

Sp1cyCarry commented now
The button makes your program exit in just one click!

hey world! e90a921

No conflicts with base branch
Merging can be performed automatically.

Merge pull request You can also merge this with the command line. View command line instructions.

Still in progress? Convert to draft

Add a comment

Write Preview

Add your comment here...

Markdown is supported Paste, drop, or click to add files

Close pull request Comment



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

The screenshot shows a GitHub pull request interface. At the top right, there are green (+4) and red (-1) status indicators with corresponding progress bars. Below them are sections for 'Reviewers' and 'Assignees'. The 'Reviewers' section says 'No reviews' and has a link 'Still in progress? Convert to draft'. The 'Assignees' section says 'No one—assign yourself' and includes a gear icon for settings. A large red arrow points from the text 'Assignees part is also handy to specify who did the exact feature' towards the 'Assignees' section.

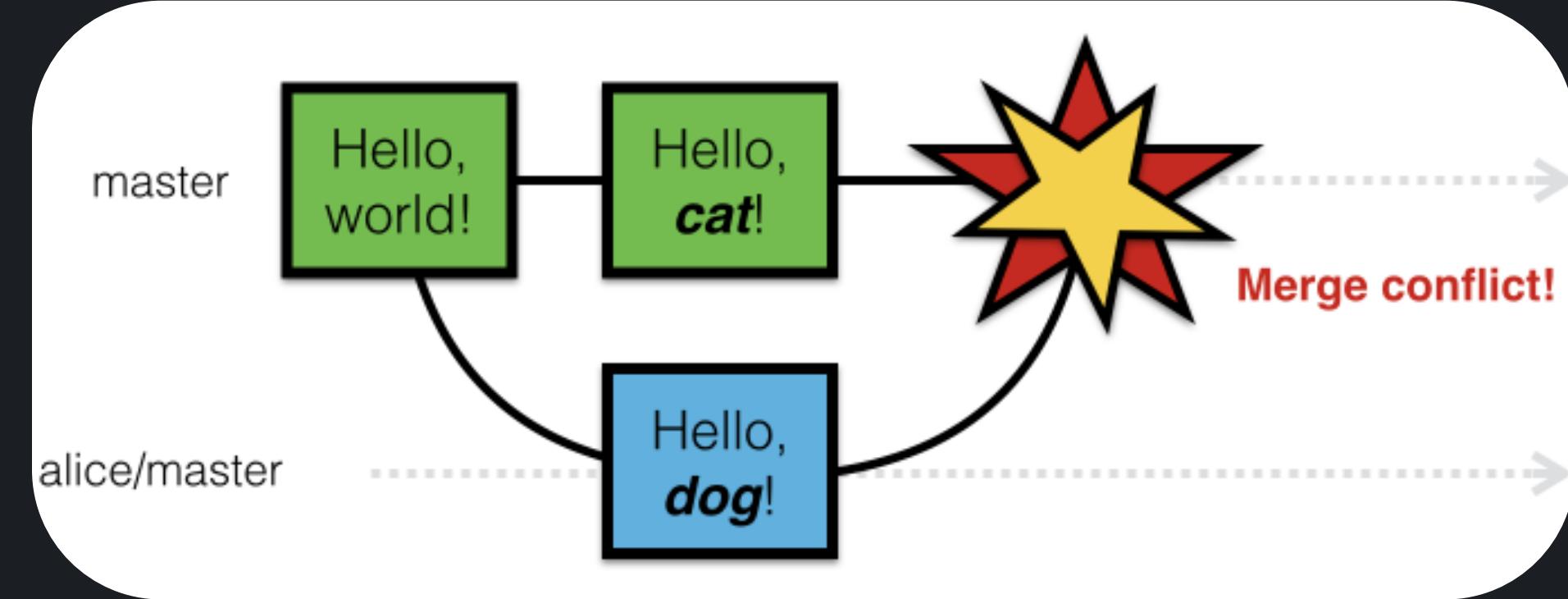
The screenshot shows a 'Reviewers' modal dialog. It has a title 'Reviewers' and a note 'Request up to 15 reviewers'. Below is a text input field with the placeholder 'type the reviewer you would assign'. A large red arrow points from the text 'Assignees part is also handy to specify who did the exact feature' towards this input field.

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



WHAT IS A MERGE CONFLICT?

Git can't decide whose version is correct



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.





beginning of conflict marker

<<<<< HEAD

identifies you

conflict divider

=====

end of conflict divider

text in version of file on GitHub

>>>>> alphanumeric hash

identifies the commit
associated with text on GitHub

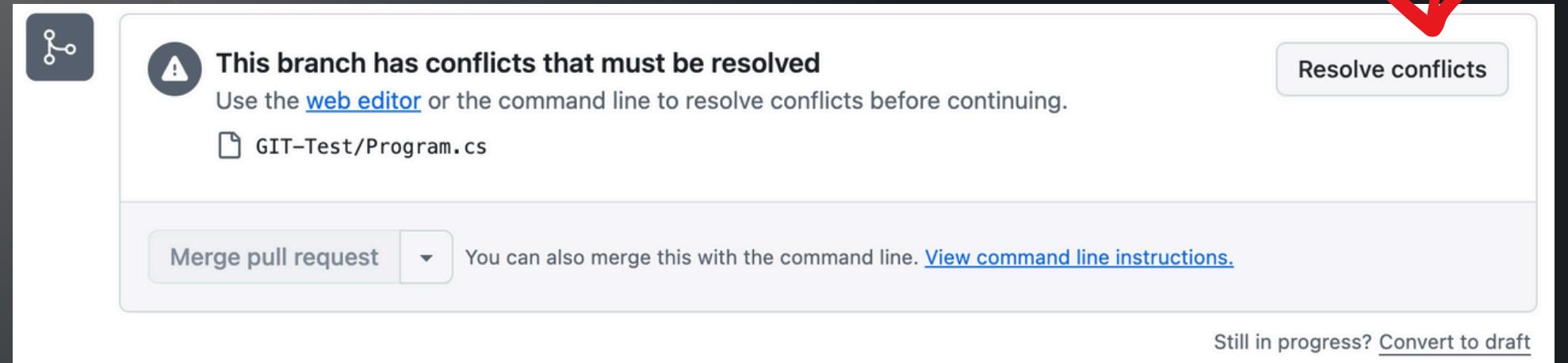
HOW TO SOLVE MERGE CONFLICTS

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.



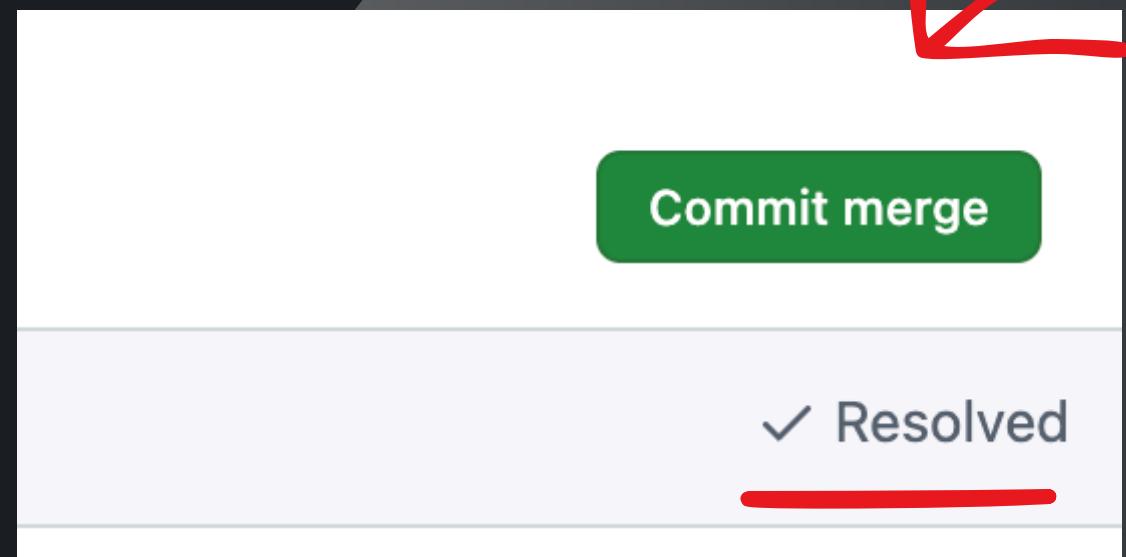
HOW TO SOLVE MERGE CONFLICTS



```
GIT-Test/Program.cs
```

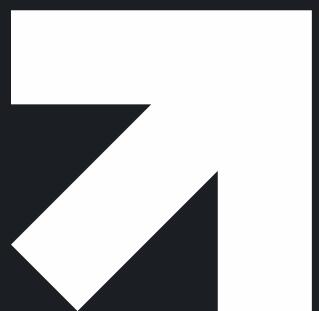
```
1  •namespace GIT_Test;
2
3  class Program
4  {
5      static void Main(string[] args)
6      {
7          Accept current change | Accept incoming change | Accept both changes
8          <<<< zaknar-merge (Current change)
9          =====
10         Console.WriteLine("Hello from Branch Zakhar!");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

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.





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.

The screenshot shows the 'Create new issue' interface on GitHub. At the top, there's a title field containing 'The edit'. Below it is a rich text editor for the description, with a 'Write' tab selected. To the right of the editor are various configuration options: 'Assignees' (No one - Assign yourself), 'Labels' (No labels), 'Projects' (No projects), and 'Milestone' (No milestone). At the bottom, there are buttons for 'Create more', 'Cancel', and a prominent green 'Create' button.



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

GIT PROJECTS

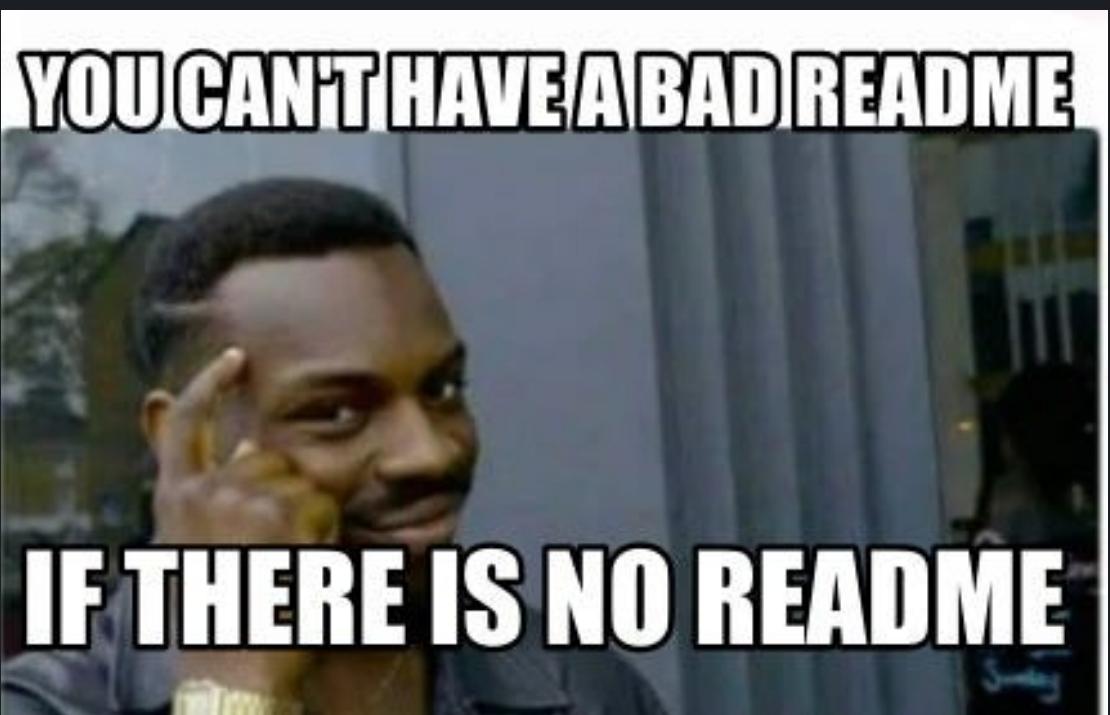
The screenshot shows a GitHub Project board titled "rauhryan/skipping_stones_repo". The board is organized into six columns: Backlog, Spec, Testing, Working, Done, and Deployed. Each column contains several cards, each representing an issue or task with a title, a number (e.g., #100, #109, #95), and a small icon. The "Backlog" column has cards like "New one expect message for create new" (#100) and "Try it again, with a relay event" (#109). The "Spec" column has cards like "100 issues?" (#104) and "Paste in a very large body" (#104). The "Testing" column has cards like "New after an archive" (#84) and "Ready, bug, dup" (#40). The "Working" column has cards like "Another new message for slack" (#103) and "Slack formatting" (#112). The "Done" column has cards like "Issue with some labels" (#43) and "This new issue should be pushed to gitter" (#73). The "Deployed" column has cards like "Hello" (#111) and "Another one with a large body" (#105). On the left side of the board, there are filters for Members, Assignment, Assigned to me, Assigned to others, Unassigned issues, Milestones, and Labels. There is also a sidebar with various project-related links and a "Create new issue" button.



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