

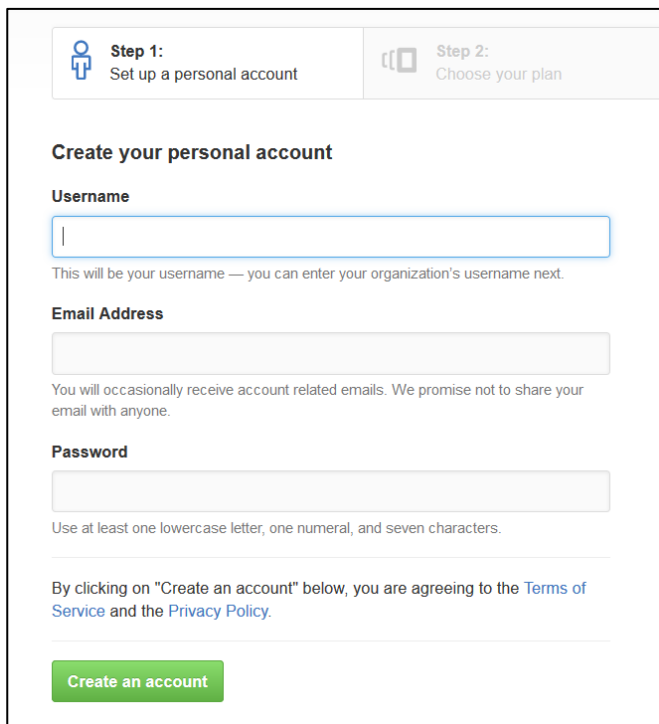
Exercises: Git and GitHub

Problems for exercises and homework for the ["Software Technologies" course @ SoftUni](#).

I. Create a GitHub Developer Profile

1. Register a GitHub Profile

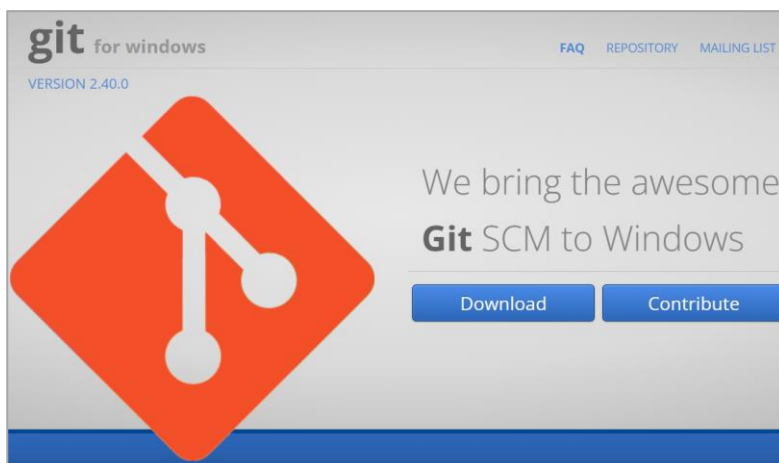
Register for a free **developer** account at GitHub: <http://github.com/>



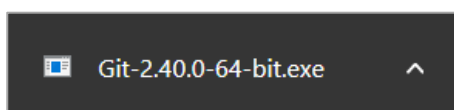
The screenshot shows the GitHub registration process. At the top, there are two steps: 'Step 1: Set up a personal account' (active) and 'Step 2: Choose your plan'. Below this, the heading 'Create your personal account' is followed by three input fields: 'Username', 'Email Address', and 'Password'. The 'Username' field has a hint: 'This will be your username — you can enter your organization's username next.' The 'Email Address' field has a hint: 'You will occasionally receive account related emails. We promise not to share your email with anyone.' The 'Password' field has a hint: 'Use at least one lowercase letter, one numeral, and seven characters.' Below the fields, there is a line of text: 'By clicking on "Create an account" below, you are agreeing to the [Terms of Service](#) and the [Privacy Policy](#).' At the bottom, there is a green button labeled 'Create an account'.

2. Install Git

1. Navigate to <https://gitforwindows.org/> and click **Download**



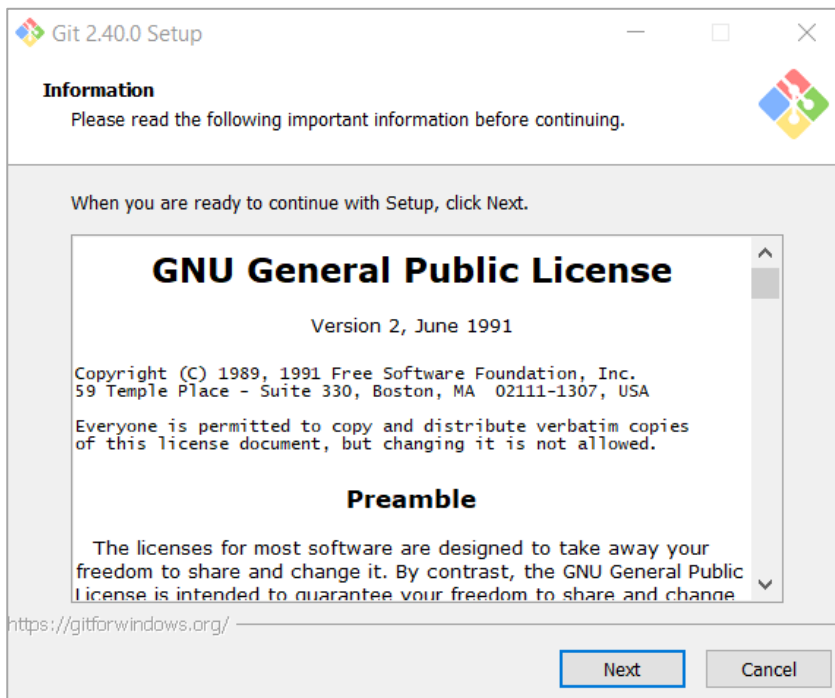
2. The .exe file will be downloaded to your browser.



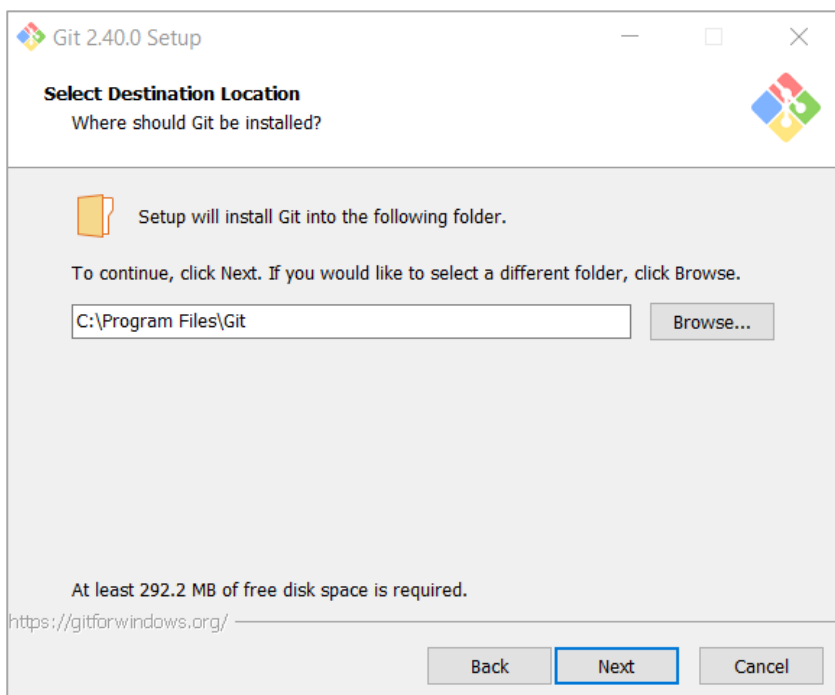
If you need the 32-bit version, go to:

<https://github.com/git-for-windows/git/releases/download/v2.40.0.windows.1/MinGit-2.40.0-32-bit.zip>

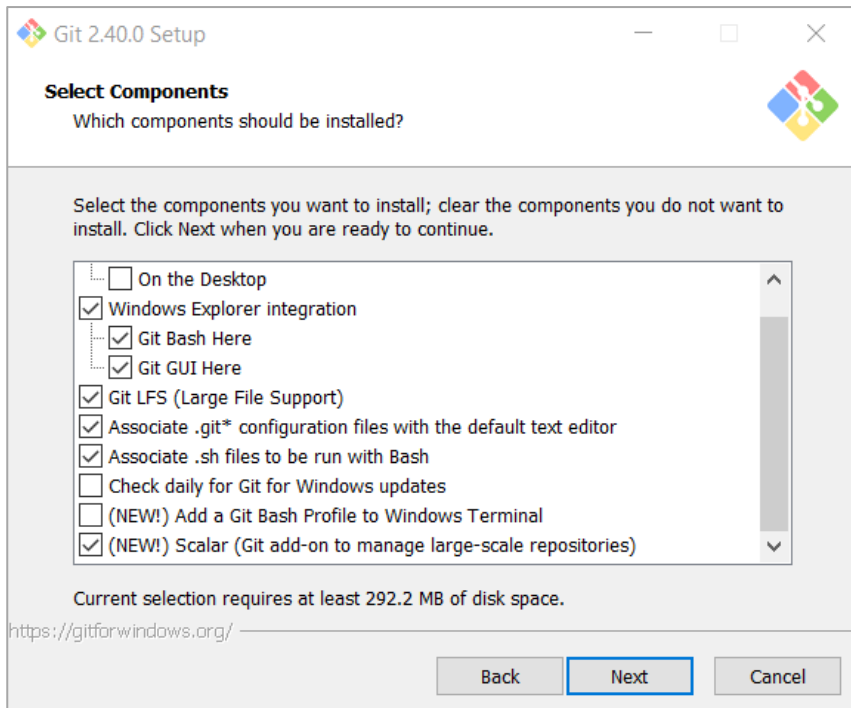
3. Double click on .exe file to initiate the installation process. When prompted by Windows, permit to make changes. You will get a General Public License Information screen. Click the "Next" button.



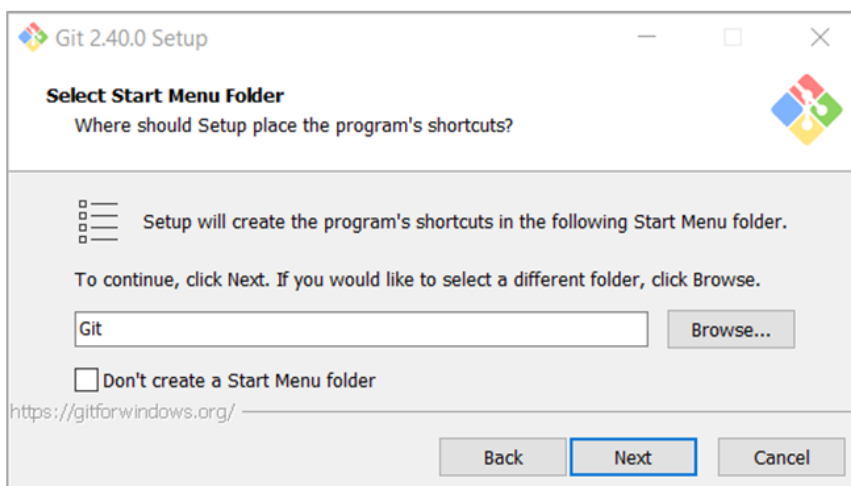
4. Choose the desired path where you want to install Git.



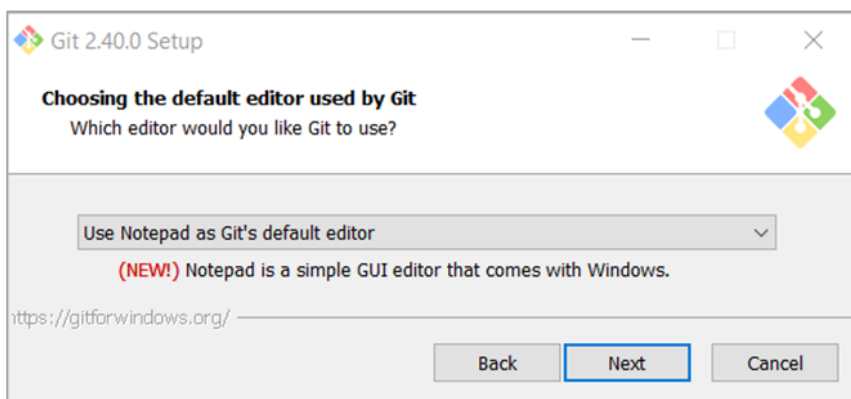
5. Leave the Select Components screen as it is. **Don't change anything.** Click **Next**.



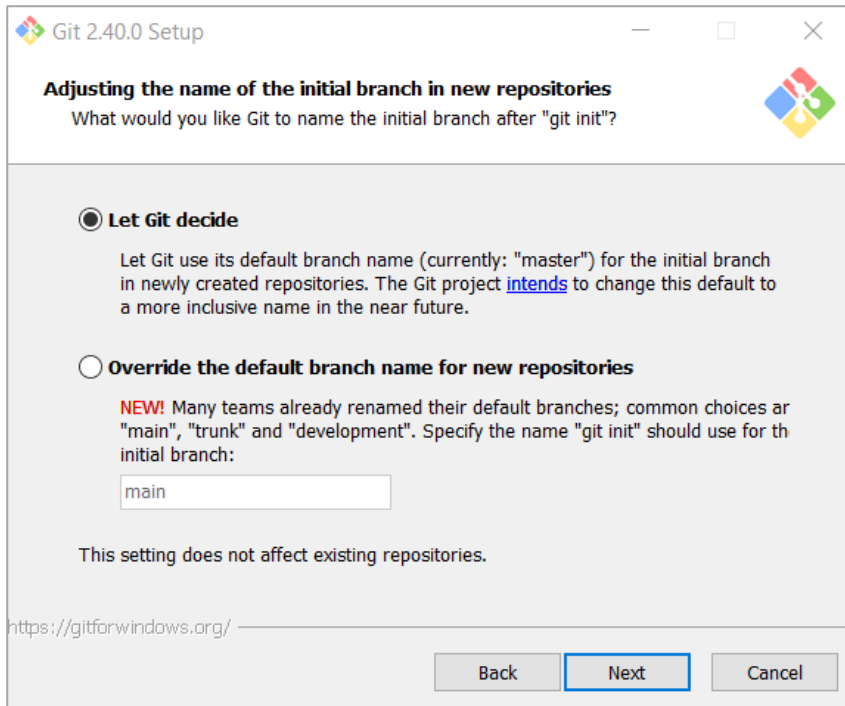
6. Select Start Menu Folder or not to create Start Menu Folder



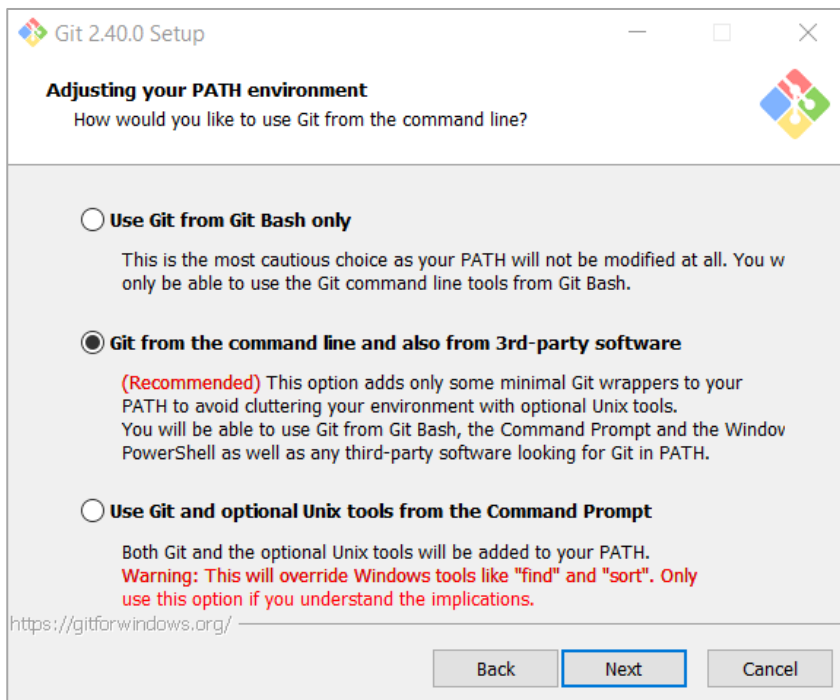
7. Choose Notepad or Notepad++ for the default editor.



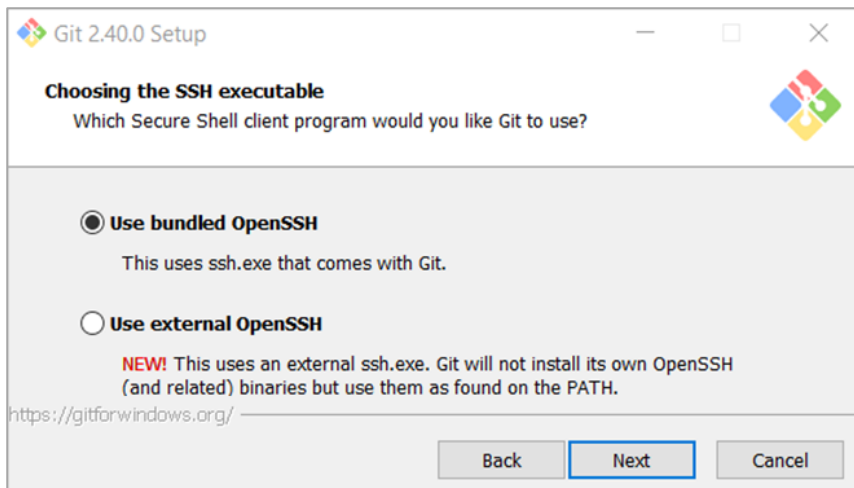
8. Let Git decide on the initial branch.



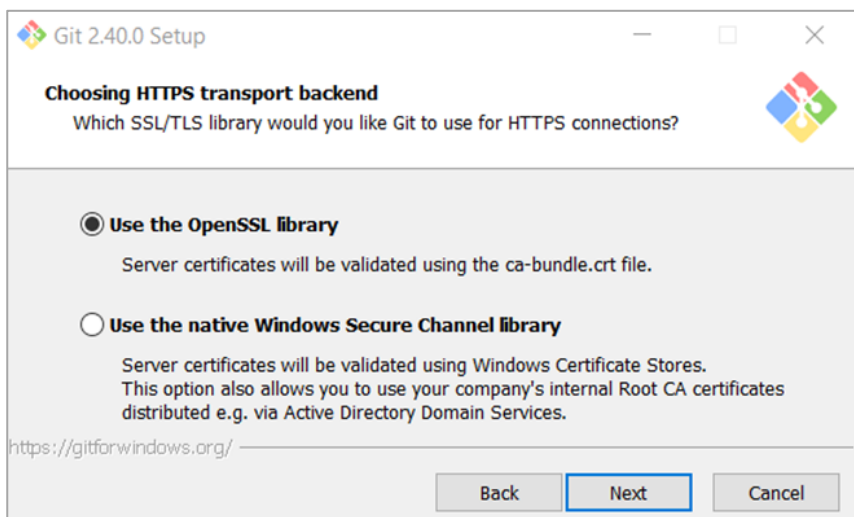
9. Choose the second option to be able to use Git from Command Prompt.



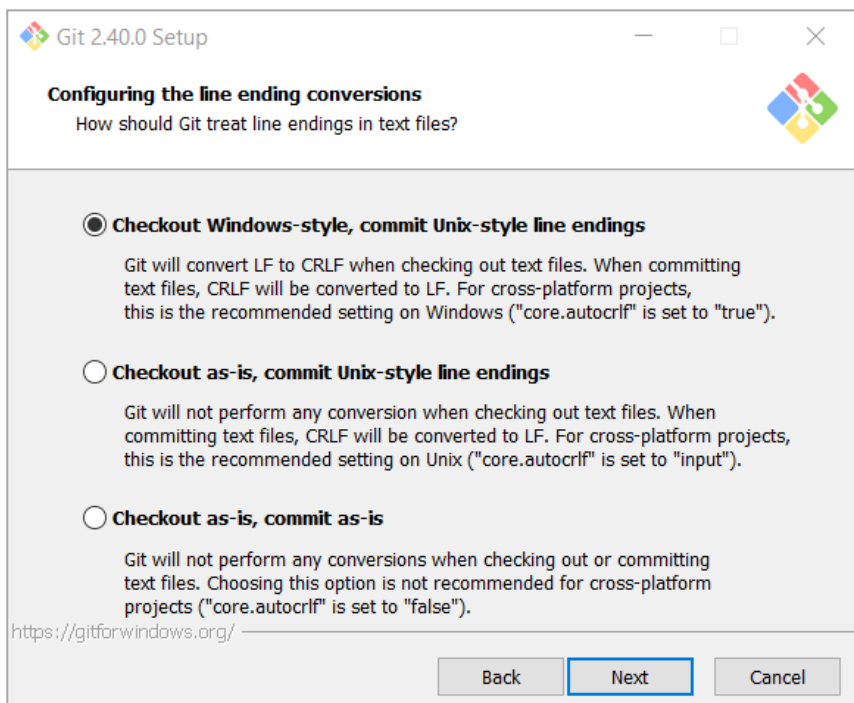
10. Leave this option as it is.



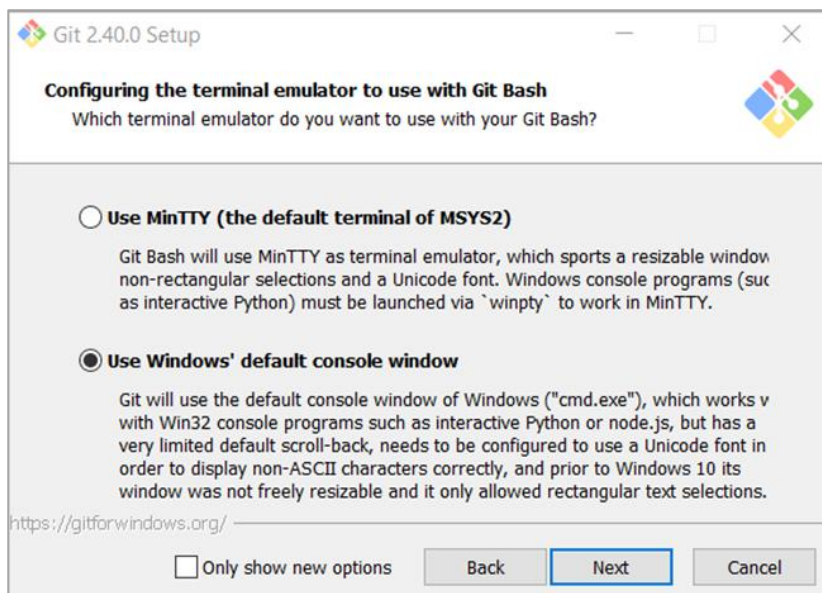
11. Leave this option as it is as well.



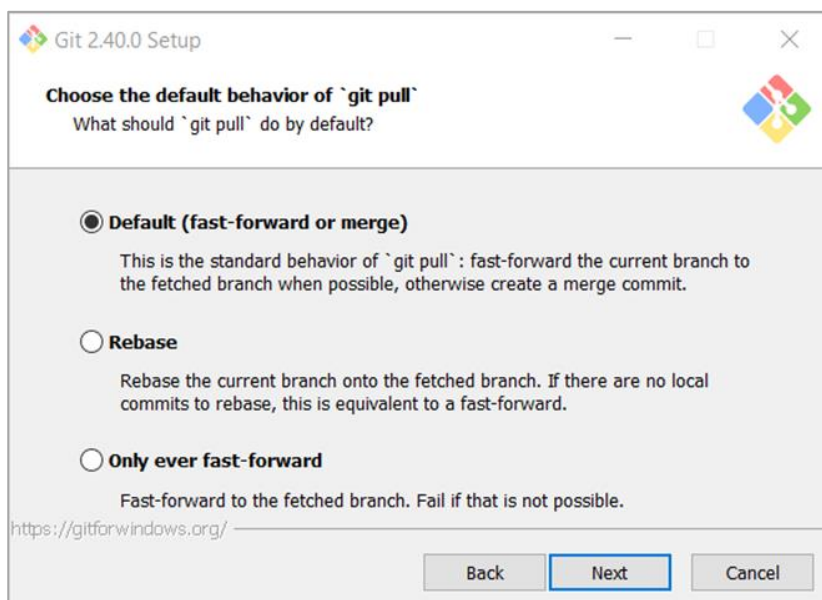
12. This one stays the same as well.



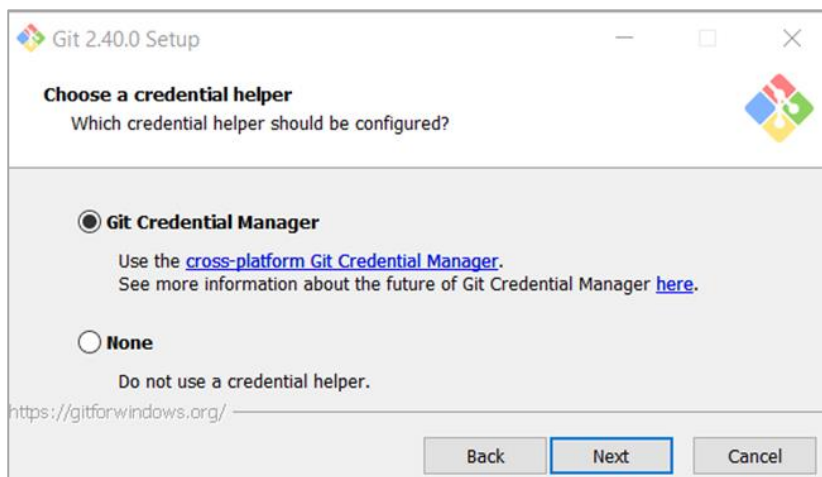
13. Select **Use Windows' default console window** option.



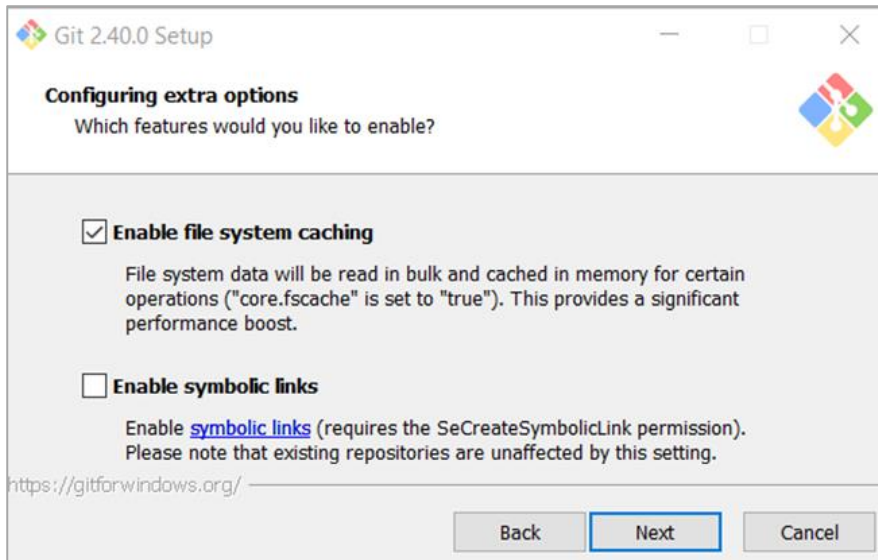
14. Default behavior – **Default**.



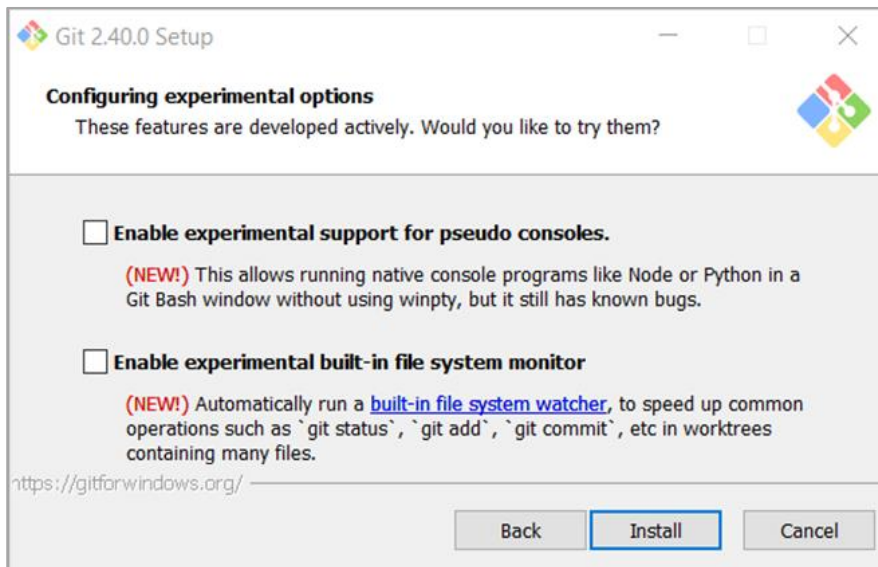
15. Git Credential Manager



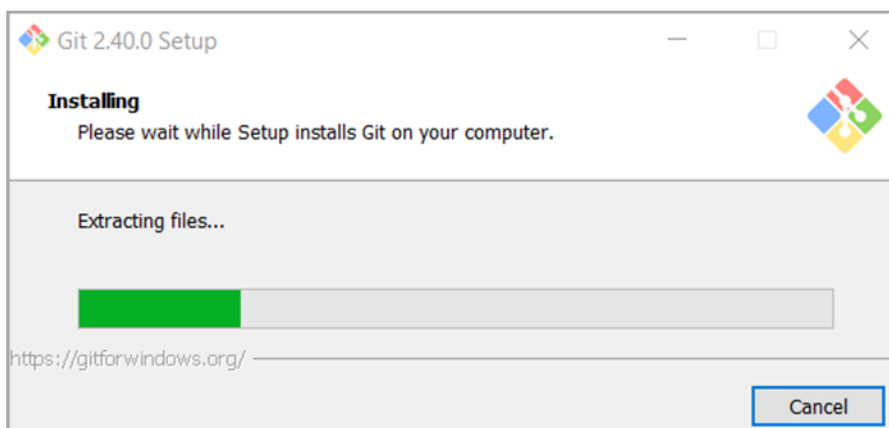
16. Enable file system caching



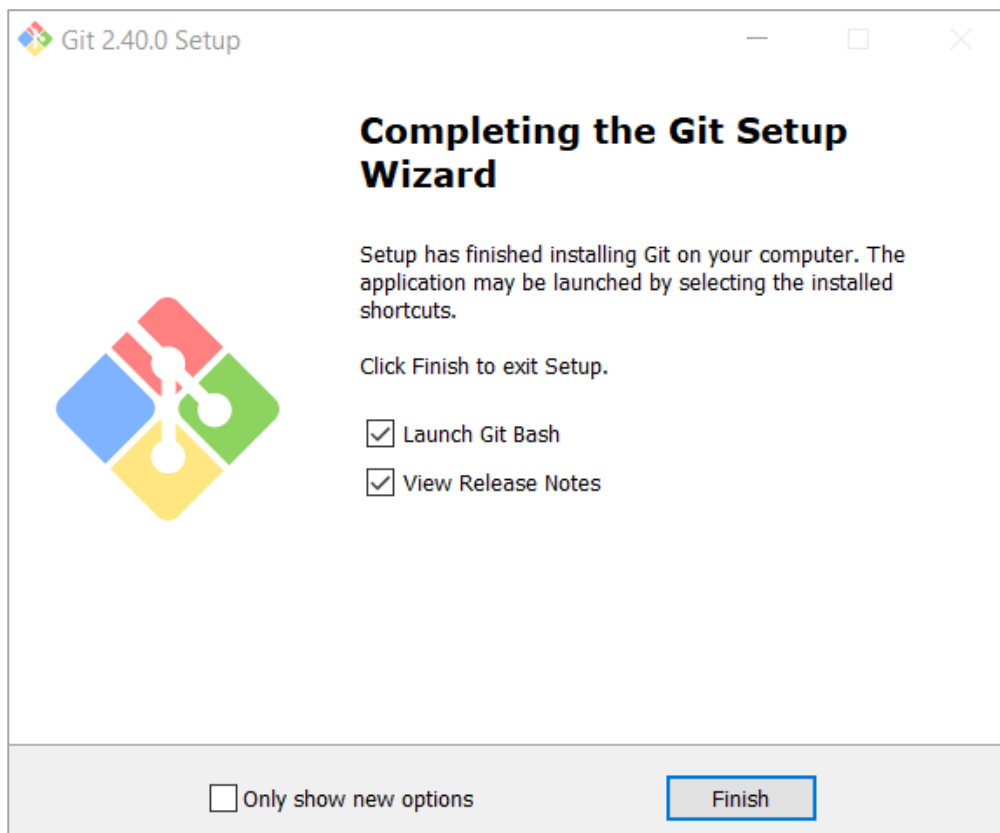
17. Leave the experimental options unchecked. Click Install.



18. Installing...



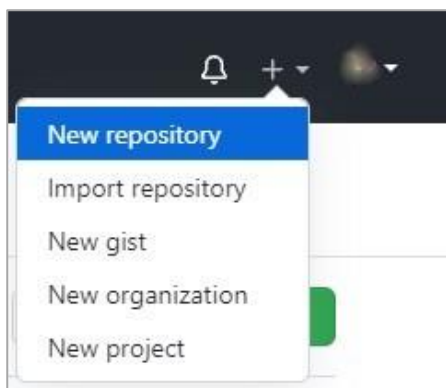
19. Complete the setup by pressing **Finish**.



II. Create a GitHub Repo and Upload Your SoftUni Projects

1. Create a GitHub Repo

In the upper-right corner of any page, use the drop-down menu, and select **New Repository**.



We choose a name according to the topic of our project. We can do it public or private.

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere?

[Import a repository.](#)

Owner *



Repository name *

SoftUni-Courses



Great repository names are short and memorable. Need inspiration? How about [miniature-spoon?](#)

Description (optional)

Courses from my education @ Softuni.



Public

Anyone on the internet can see this repository. You choose who can commit.



Private

You choose who can see and commit to this repository.

Select Initialize this repository with a **README** and click on **Create repository**.

Initialize this repository with:

☒ Add a README file

This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore

.gitignore template: None

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

Choose a license

License: None

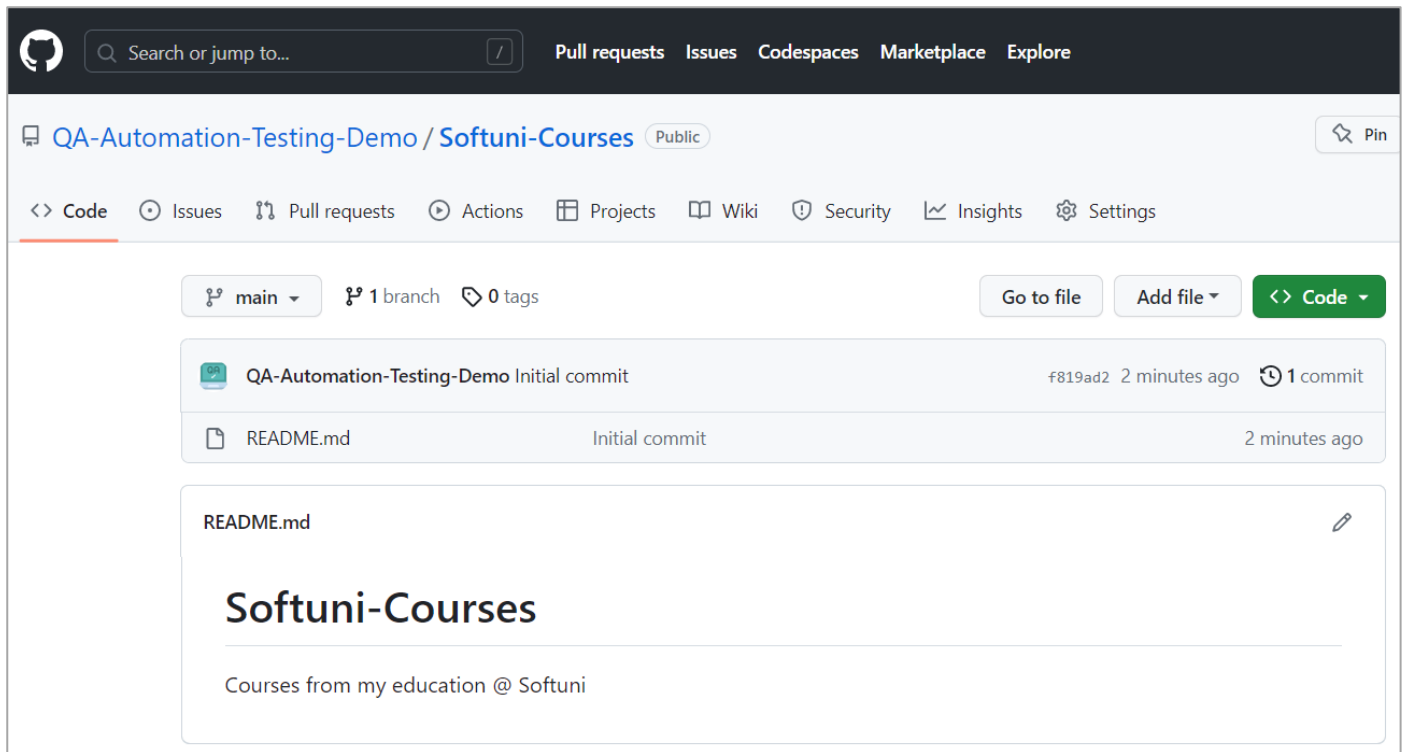
A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

This will set  main as the default branch. Change the default name in your [settings](#).

 You are creating a public repository in your personal account.

Create repository

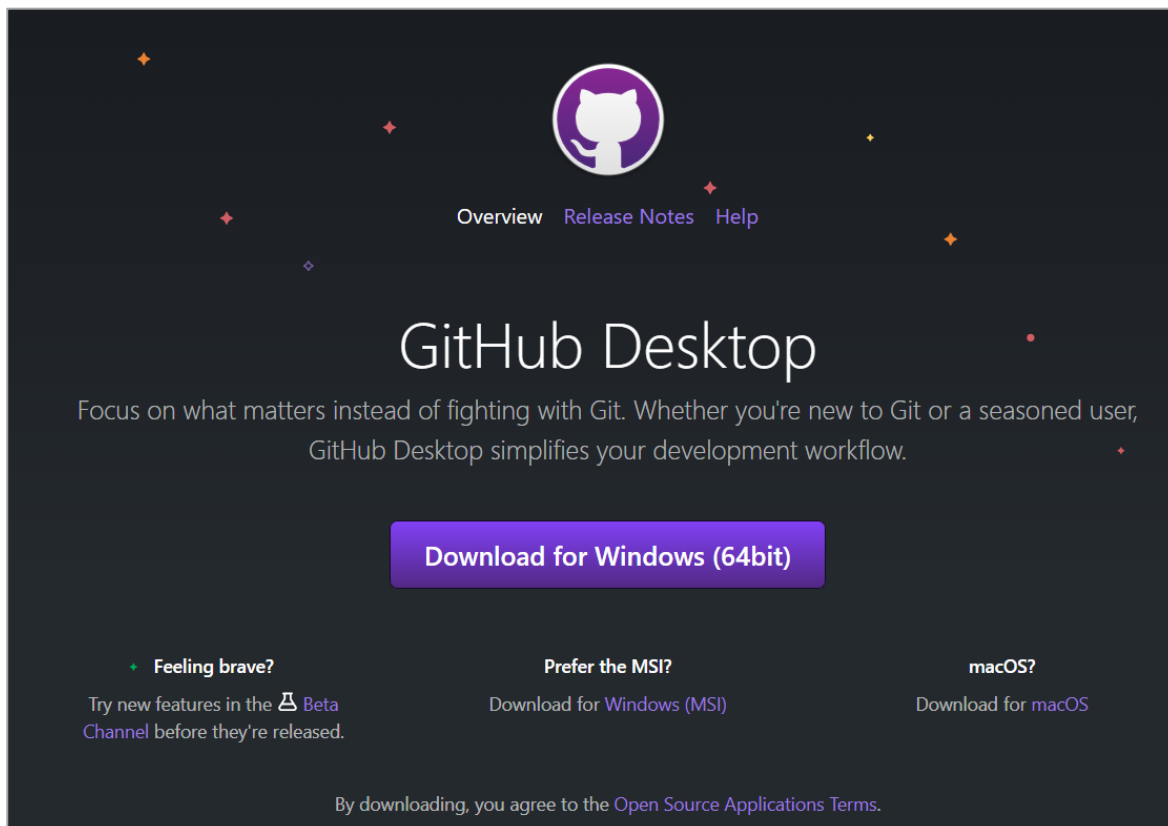
Your repository is now created.



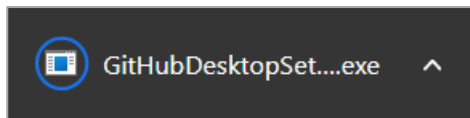
2. Download and install GitHub Desktop

Navigate to <https://desktop.github.com/>

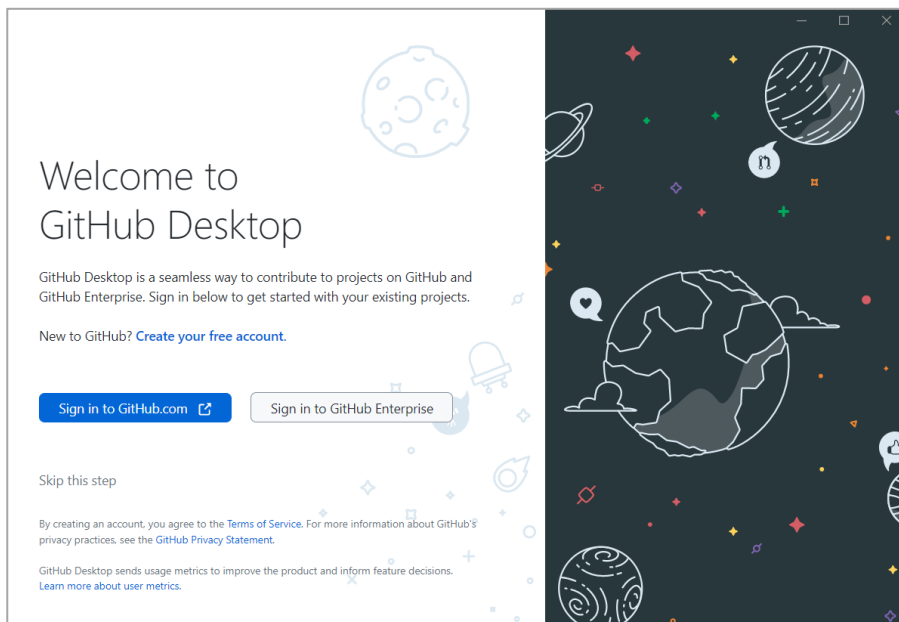
For this tutorial, we're downloading GitHub Desktop for Windows (64bit).



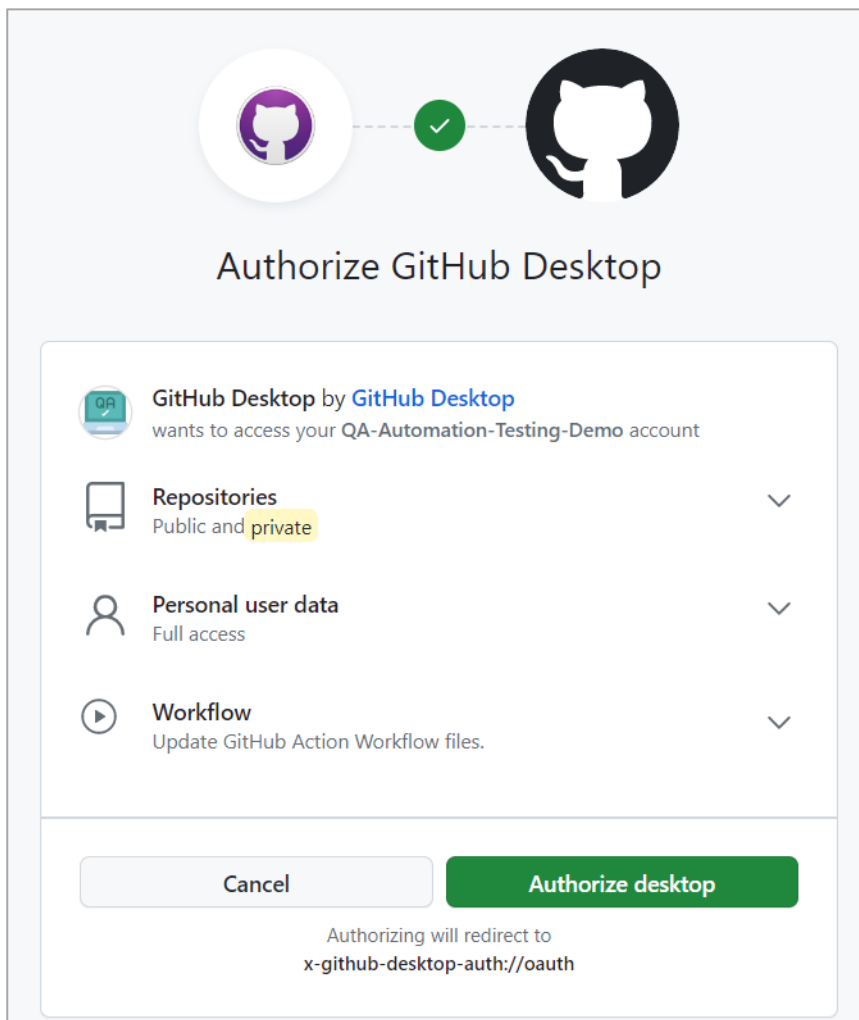
After starting the installation file...



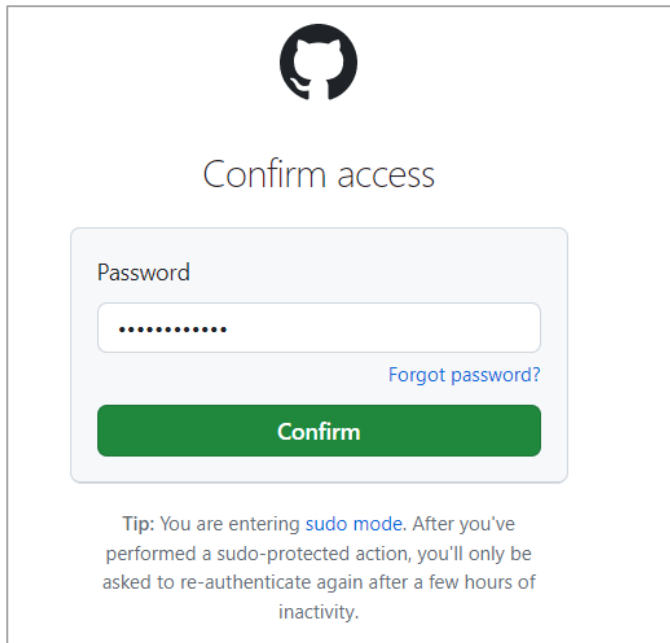
You will be greeted by the welcome screen. Since you already have an account, hit the "Sign in to GitHub.com" button



A new pop-up window will appear in your browser asking you to authorize GitHub Desktop.

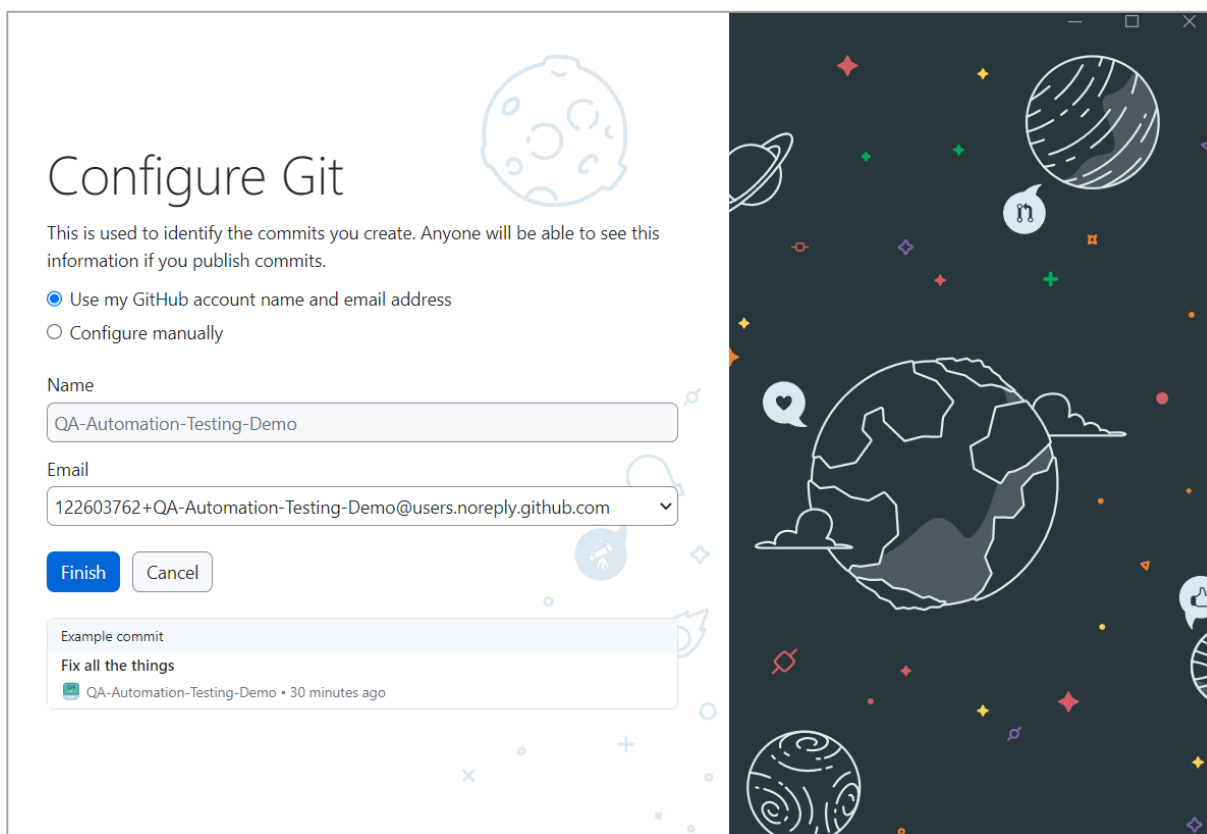


And then confirm access with your GitHub password.



The image shows the GitHub 'Confirm access' screen. At the top is the GitHub logo. Below it, the text 'Confirm access' is centered. There is a password input field with a masked password '.....'. To the right of the input field is a link that says 'Forgot password?'. Below the input field is a green button labeled 'Confirm'. At the bottom, there is a tip: 'Tip: You are entering **sudo mode**. After you've performed a sudo-protected action, you'll only be asked to re-authenticate again after a few hours of inactivity.'

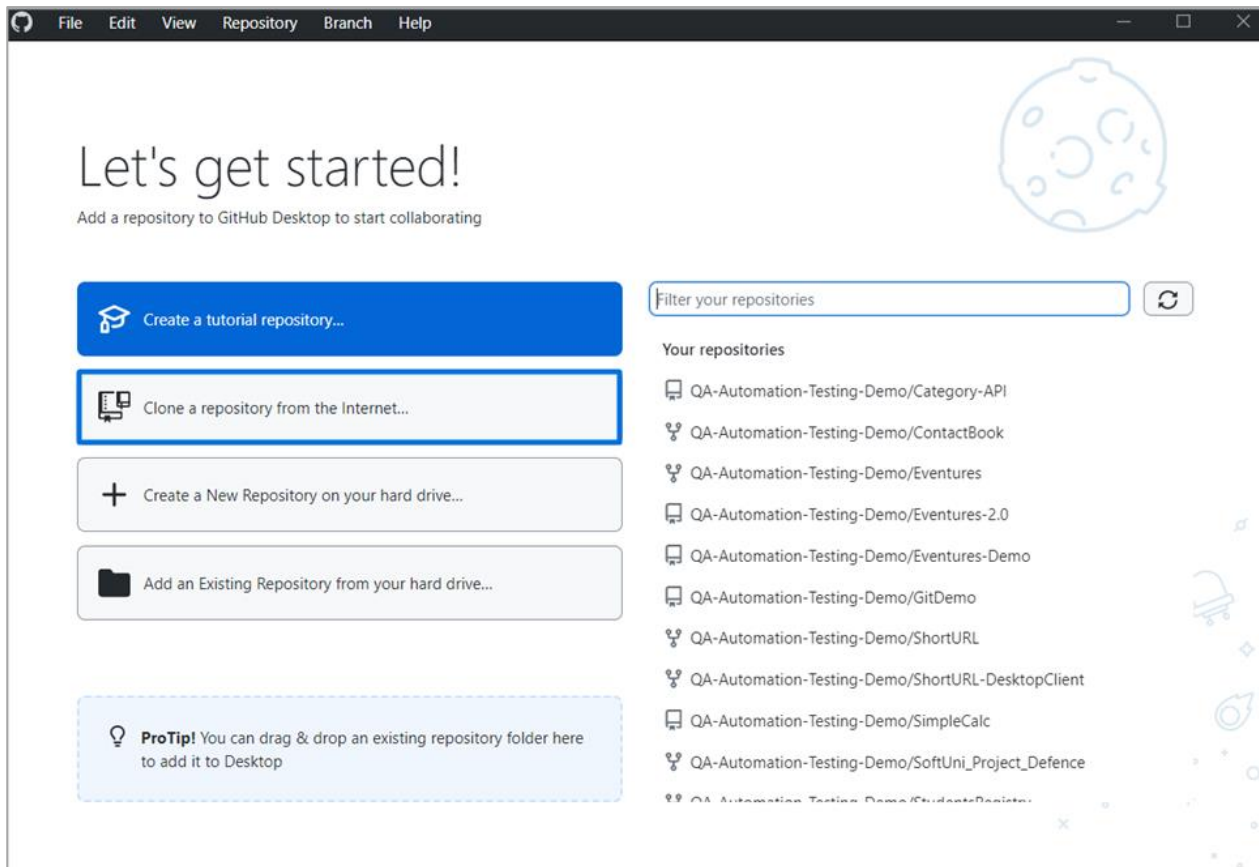
If the authorization is successful, you will be prompted to configure git. Leave it as it is and hit "Finish".



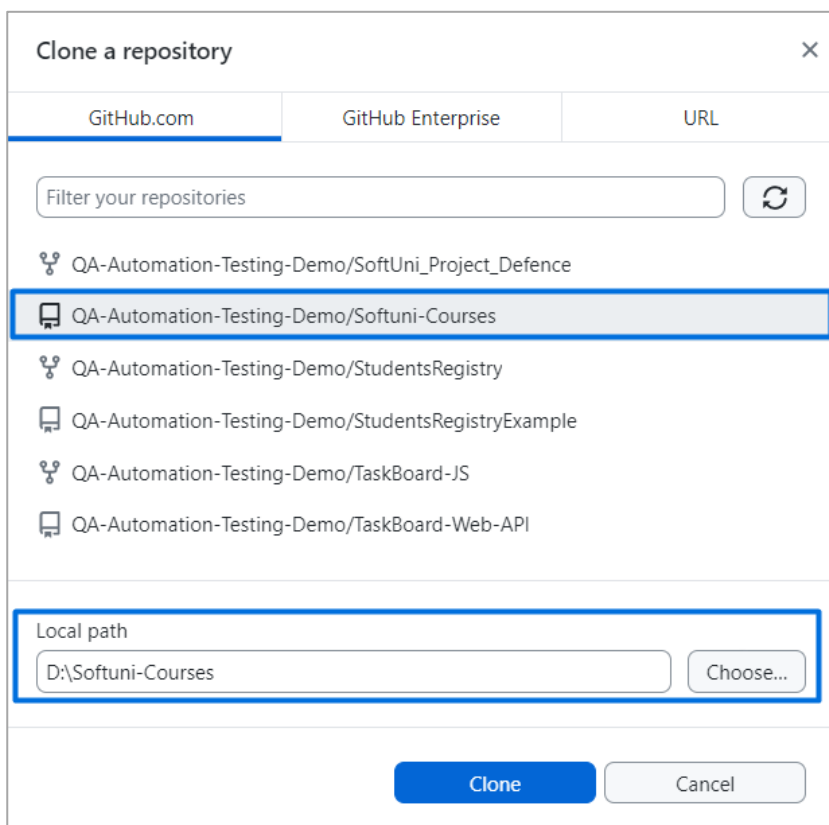
The image shows the 'Configure Git' screen. On the left, there is a form with the title 'Configure Git' and a subtitle 'This is used to identify the commits you create. Anyone will be able to see this information if you publish commits.' There are two radio buttons: 'Use my GitHub account name and email address' (selected) and 'Configure manually'. Below these are input fields for 'Name' (QA-Automation-Testing-Demo) and 'Email' (122603762+QA-Automation-Testing-Demo@users.noreply.github.com). There are 'Finish' and 'Cancel' buttons. At the bottom, there is an 'Example commit' section showing 'Fix all the things' by 'QA-Automation-Testing-Demo • 30 minutes ago'. On the right, there is a decorative space-themed illustration with planets, stars, and a rocket.

3. Clone a repository

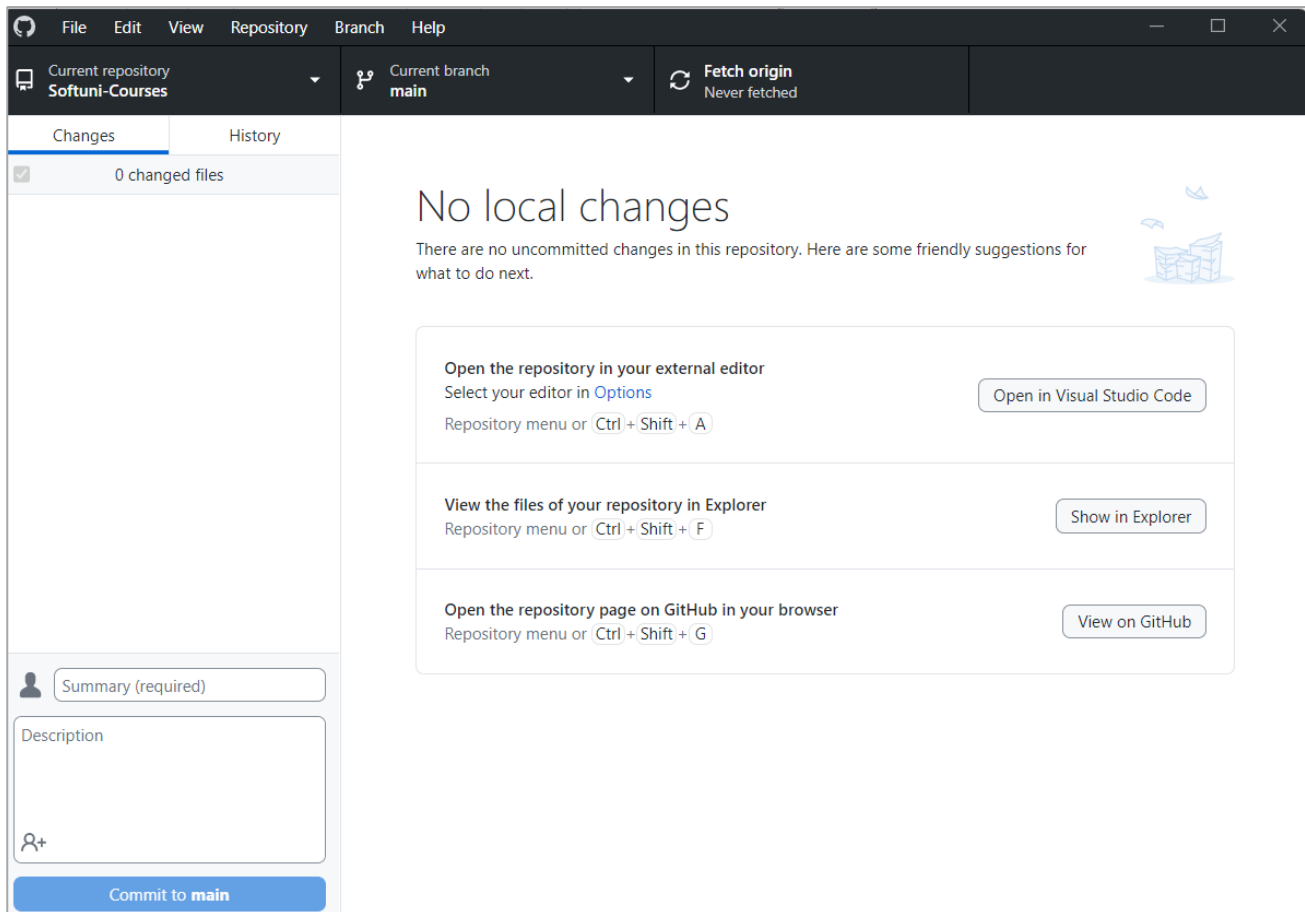
So, now your GitHub Desktop is connected to your GitHub account and you can clone the repository that you just created on your local machine. Select "Clone a repository from the Internet...".



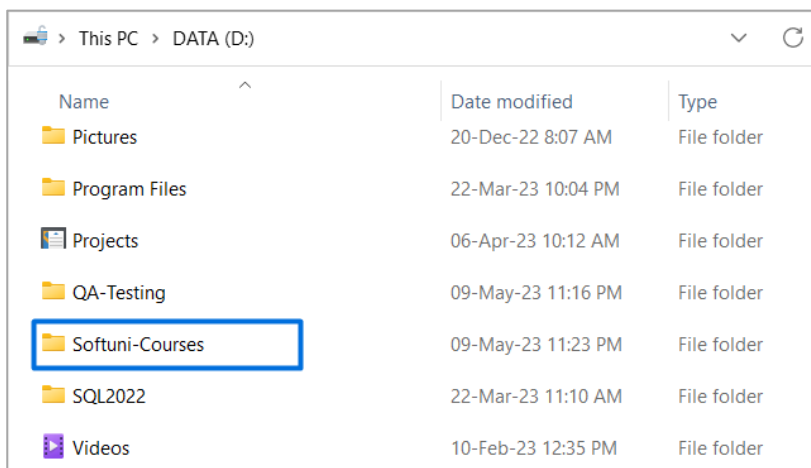
Then choose the repository that you created (The one called "Softuni-Courses"). Choose the path where you would like to save it on your local machine and click "Clone".



The repository is now cloned locally.

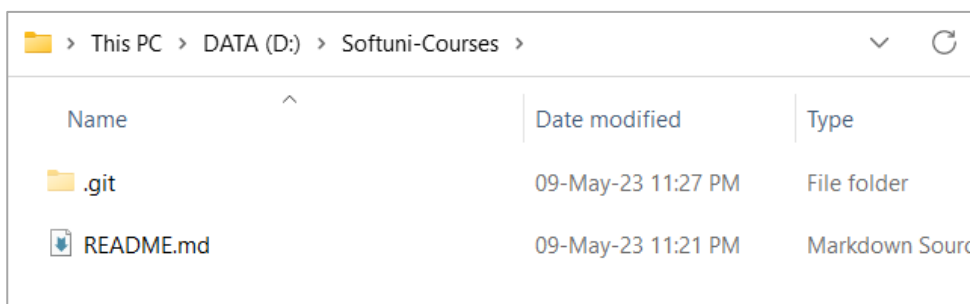


You can go and check if the new folder is properly created locally.



4. Commit and Push

Now, open the "Softuni-Courses" folder on your local machine. This is how it should look like.

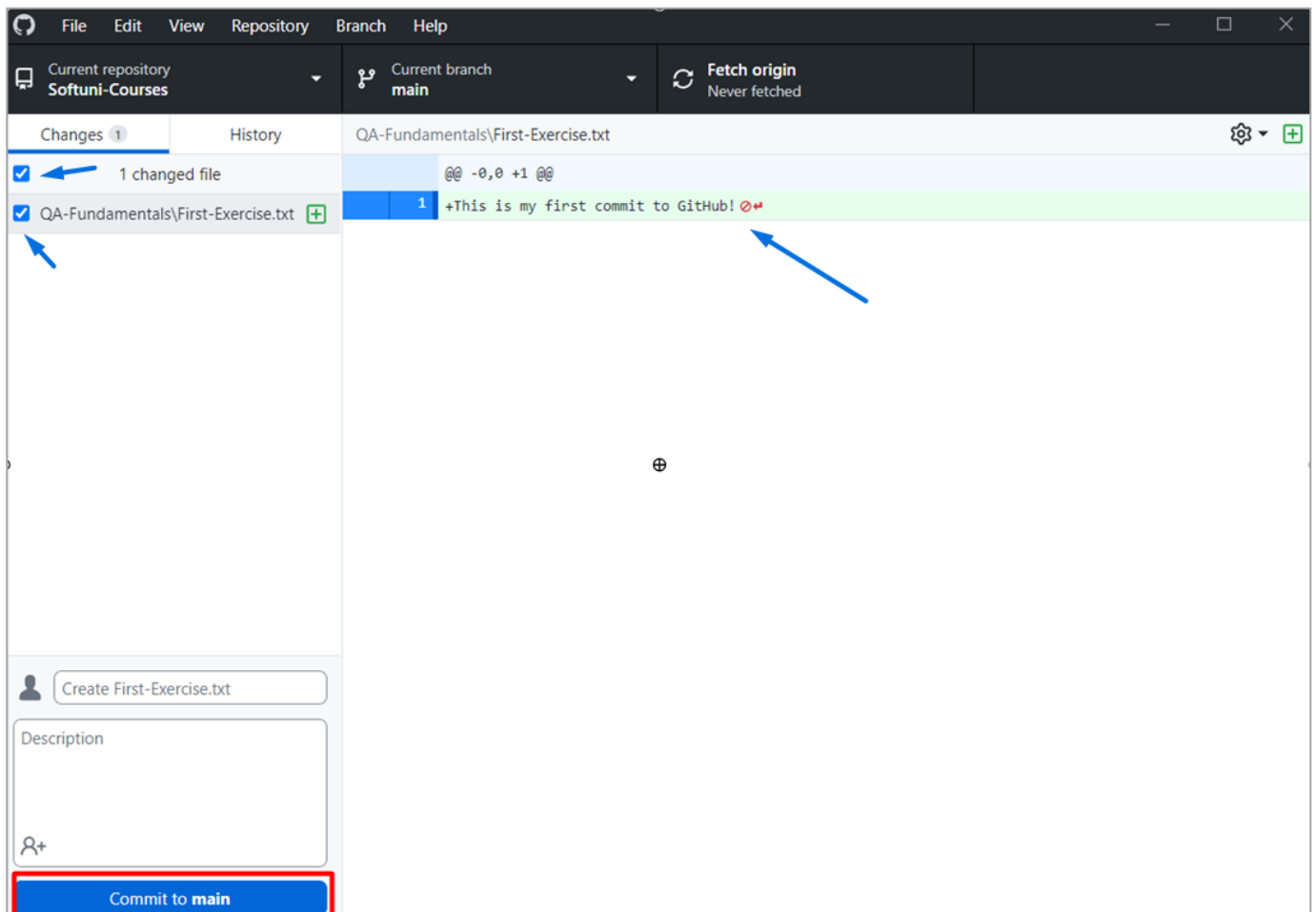


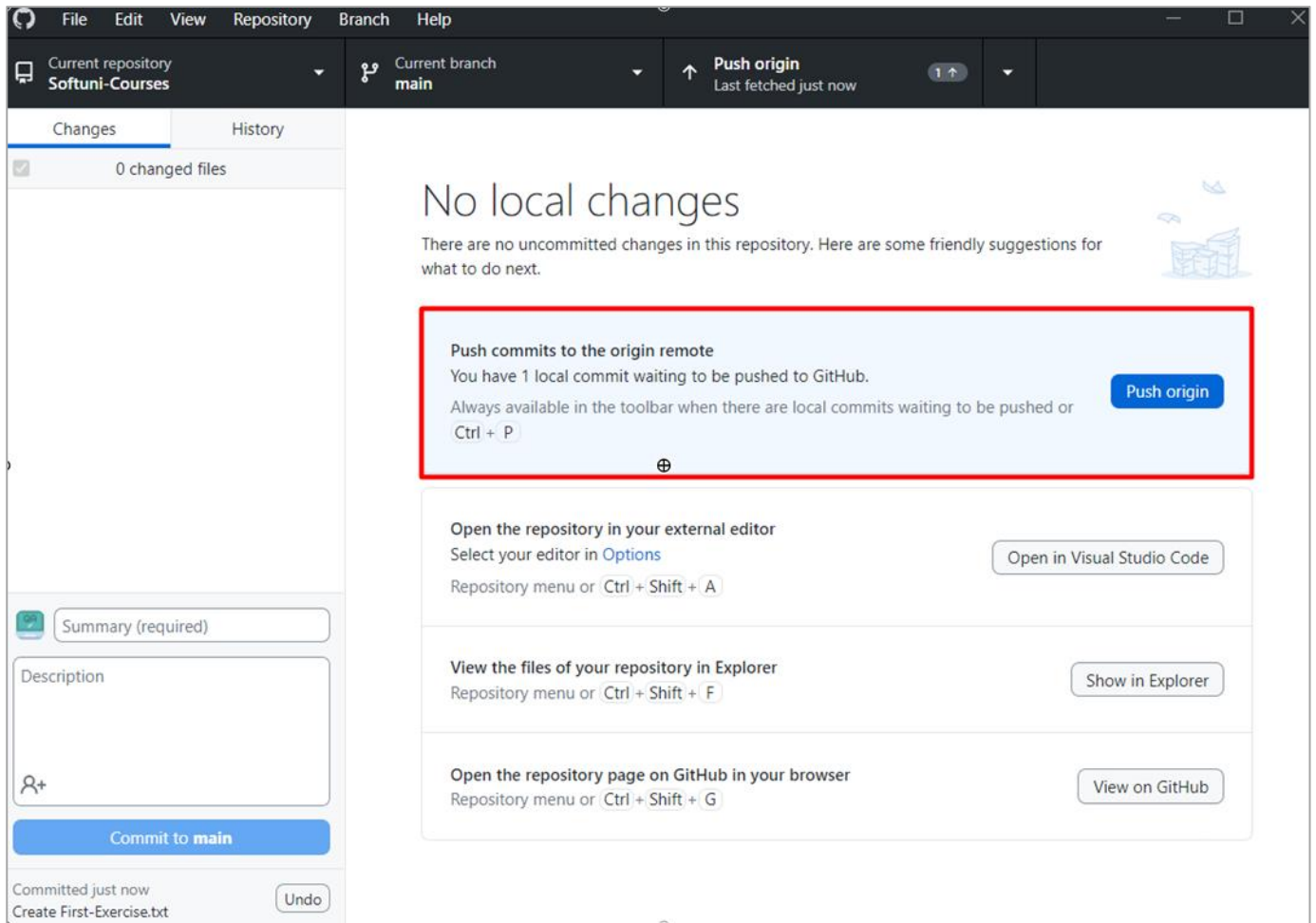
Create a new folder inside and call it "QA-Fundamentals". And inside the "QA-Fundamentals" folder, create a new text file (.txt) with an editor of your choosing and write something in it. In this case, our file is called "First-Exercise" and the text in it reads: "This is my first commit to GitHub!".

This PC > DATA (D:) > Softuni-Courses >		
Name	Date modified	Type
.git	09-May-23 11:27 PM	File folder
QA-Fundamentals	10-May-23 12:06 PM	File folder
README.md	09-May-23 11:21 PM	Markdown Source

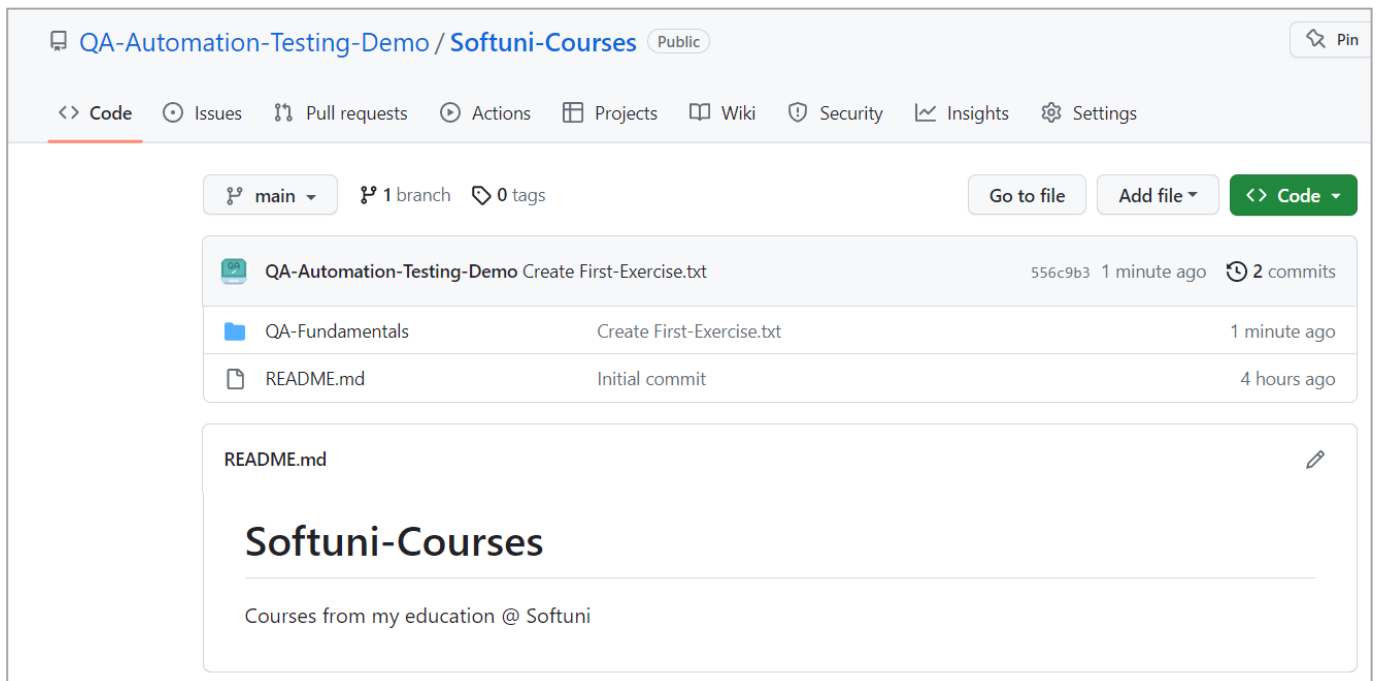
This PC > DATA (D:) > Softuni-Courses > QA-Fundamentals		
Name	Date modified	Type
First-Exercise.txt	09-May-23 11:25 PM	Text Document

Now, head back to GitHub desktop. As you can see there is one uncommitted change that is pending. Hit the **"Commit to main"** button and then **"Push origin"**.





Head back to GitHub and check if the changes that you just made are present.



III. Git Bash

1. Check Git version

To **check the Git version**, run the following command:

- from the command prompt **git version**
- from git bash **\$ git -version**

2. Check Credentials

To **check if Git is configured** on your local machine, you can use the following command in the Git console :

```
$ git config --list
```

This command will display the Git configuration settings that are currently set. If Git is properly configured, you should see output similar to the following:

```
user.name=Your Name
```

```
user.email=your-email@example.com
```

The **user.name** and **user.email settings** indicate that Git has been configured with your name and email address, which are important for identifying your commits.

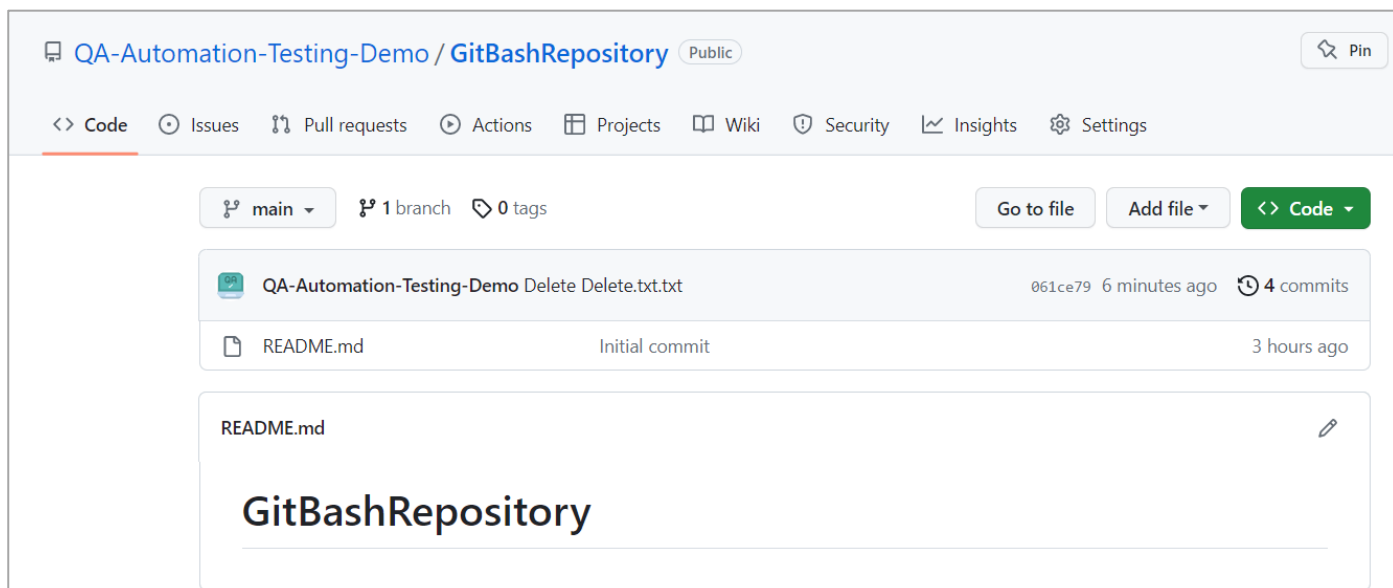
If you don't see any output or the output does not include the **user.name** and **user.email settings**, it means that Git is **either not installed or not configured on your machine**. In that case, you can follow the installation instructions for your operating system and **then configure Git by running the following commands**:

```
$ git config --global user.name "Your Name"
```

```
$ git config --global user.email "your-email@example.com"
```

3. Clone GitHub repository

Since you already know how to create repositories on GitHub, go and create a new one.

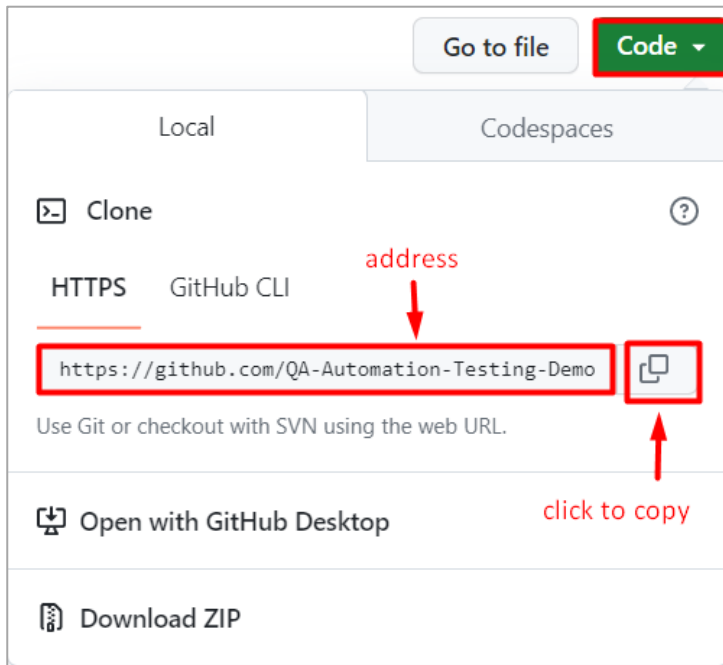


For the following exercise, you can use the **command prompt** or **git bash**. The syntax is very similar. We are using GitBash, since it's colorful.

To **change directories**, use the command **cd** followed by the **name of the directory**. In our case we would like to clone the repository on **drive D**, so we're navigating to D:/ by typing the following command:

```
$ cd d:/
```

To clone the repository on your local machine you will need its address.



Type the following command:

```
$ git clone <address-of-your-repo>
```

This is how it should look like:

```
MINGW64:/d
@LAPTOP-OQA6P0JQ MINGW64 ~ (master)
$ cd d:/

@LAPTOP-OQA6P0JQ MINGW64 /d
$ git clone https://github.com/QA-Automation-Testing-Demo/GitBashRepository.git
Cloning into 'GitBashRepository'...
remote: Enumerating objects: 9, done.
remote: Counting objects: 100% (9/9), done.
remote: Compressing objects: 100% (6/6), done.
remote: Total 9 (delta 0), reused 3 (delta 0), pack-reused 0
Receiving objects: 100% (9/9), done.

@LAPTOP-OQA6P0JQ MINGW64 /d
$ |
```

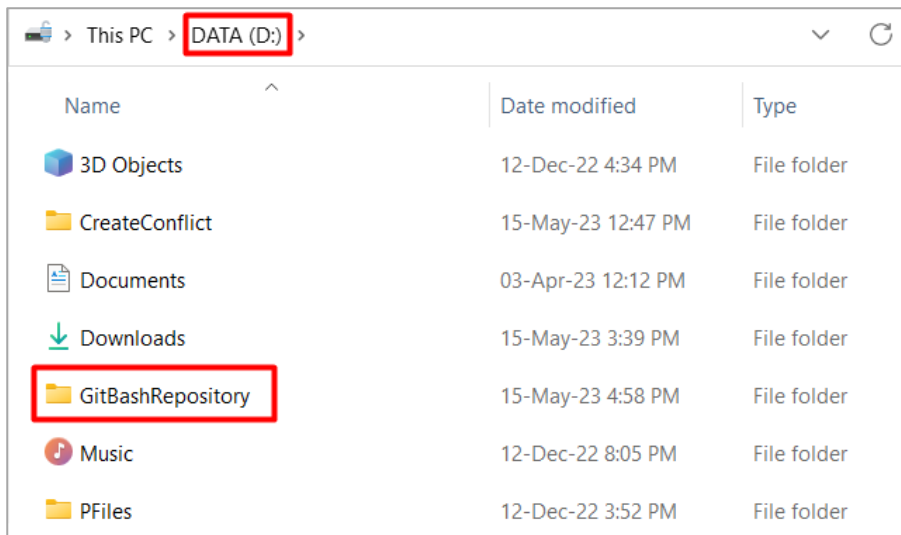
Navigate to the newly created directory by running the following command:

```
cd <repository-name>
```

```
mddim@LAPTOP-OQA6P0JQ MINGW64 /d
$ cd gitbashrepository

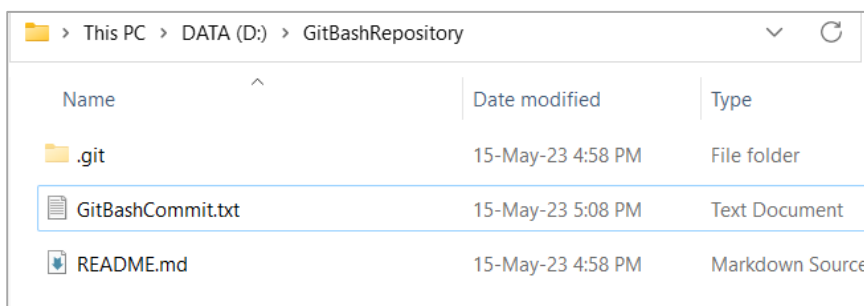
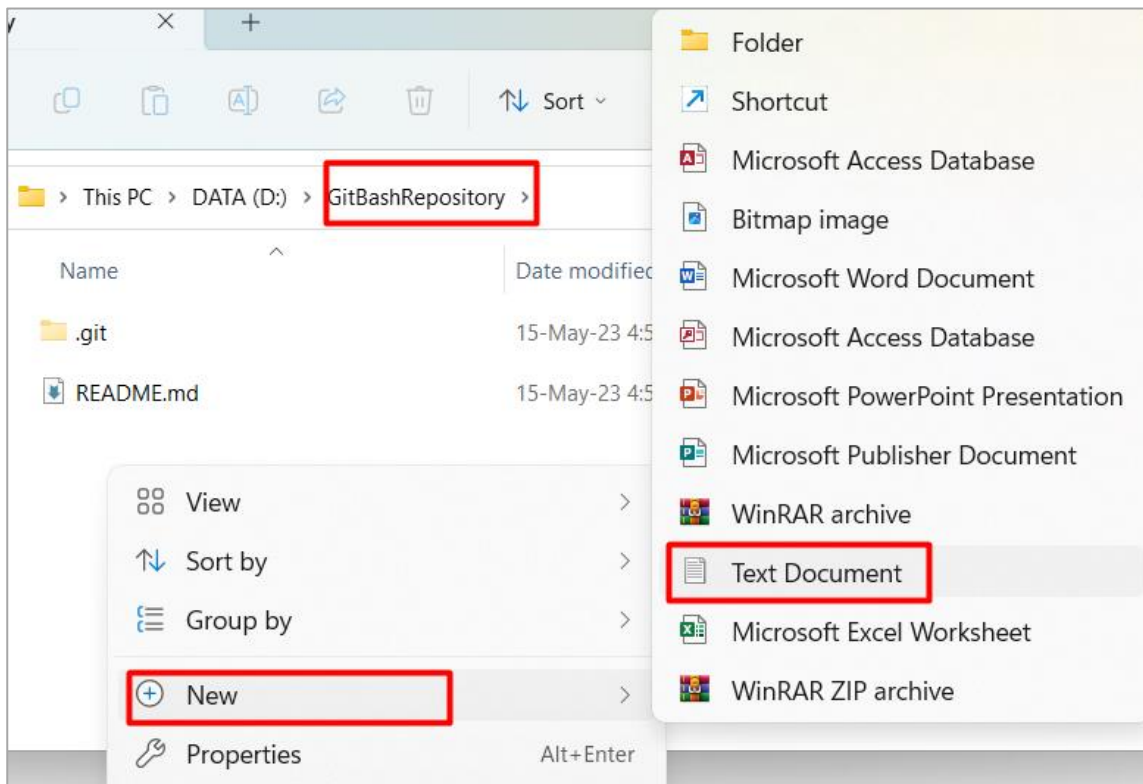
mddim@LAPTOP-OQA6P0JQ MINGW64 /d/gitbashrepository (main)
$
```

You can also check via Windows Explorer if everything is cloned on your machine.



4. Edit, Commit, and Push

Now, create a new Text document with the editor of your choosing.



Use your favorite text editor or IDE to modify files in the repository.

Add new files, delete files, or make changes to existing files.

Don't change the .git folder!

To see the status of your changes, run the following command:

\$ git status

```
@LAPTOP-OQA6P0JQ MINGW64 /d/gitbashrepository (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        GitBashCommit.txt

nothing added to commit but untracked files present (use "git add" to track)
```

Let's commit your changes to GitHub.

Stage the changes you want to commit using the following command:

\$ git add <file1> <file2> ...

Alternatively, to stage all changes, you can use:

\$ git add .

```
@LAPTOP-OQA6P0JQ MINGW64 /d/gitbashrepository (main)
$ git add GitBashCommit.txt
```

Commit the staged changes with a descriptive message:

\$ git commit -m "Commit message"

```
@LAPTOP-OQA6P0JQ MINGW64 /d/gitbashrepository (main)
$ git commit -m "First Time committing from GitBash"
[main 21f7155] First Time committing from GitBash
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 GitBashCommit.txt
```

Push your commits to the remote repository:

\$ git push origin

```
@LAPTOP-OQA6P0JQ MINGW64 /d/gitbashrepository (main)
$ git push origin
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 324 bytes | 324.00 KiB/s, done.
Total 3 (delta 0), reused 1 (delta 0), pack-reused 0
To https://github.com/QA-Automation-Testing-Demo/GitBashRepository.git
061ce79..21f7155  main -> main
```

Congratulations! You have successfully created a repository, cloned it, made changes, committed them, and pushed them back using GitBash.

IV. * Create and Resolve a Conflict

In the real world, conflicts are typically created when multiple users are working on the same project, and they try to merge their changes. This exercise simulates that by creating a conflict within the same local environment.

1. Create a conflict

On GitHub: Create a new repository, as we already taught you how to do it. We named ours “TestRepo”.

 [QA-Automation-Testing-Demo / TestRepo](#)

On GitHub Desktop: Clone the new repository to a folder of your choosing on your local machine. We named our folder TestRepo1.

On Git Bash: Clone the same repository from GitHub, but to another folder. We named ours TestRepo.

```
MINGW64:d/testrepo
P0JQ MINGW64 ~ (master)
$ cd d:
P0JQ MINGW64 /d
$ git clone https://github.com/QA-Automation-Testing-Demo/TestRepo.git
Cloning into 'TestRepo'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (3/3), done.
```

Now you have two absolutely identical copies of your repository in two different folders on your computer. Imagine that these are two different people, each working on its own copy, on its own computer.

In “TestRepo” folder create a .txt file, give it a name, and add some text inside and save the file. For example, our file is called **CommitFromGitBash.txt**.

On GitBash: Go to the cloned repository directory using: **cd path/to/your/repository**

Stage the changes using: **git add CommitFromGitBash.txt**

Commit the changes using: **git commit -m "First commit"**

Push the changes to GitHub using: **git push origin main**

On GitHub: Go and check if the file that you just pushed is uploaded.

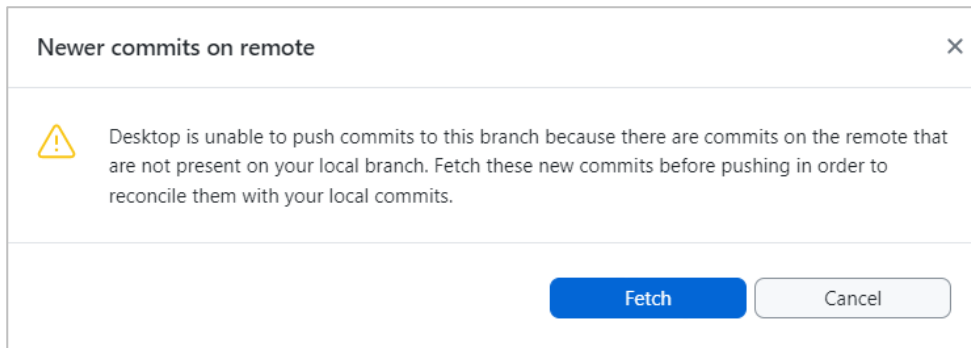


Now, in "TestRepo1" folder create another .txt file, give it a name add some text inside and save the file. For example, we named our file **CommitFromGitHubDesktop.txt**.

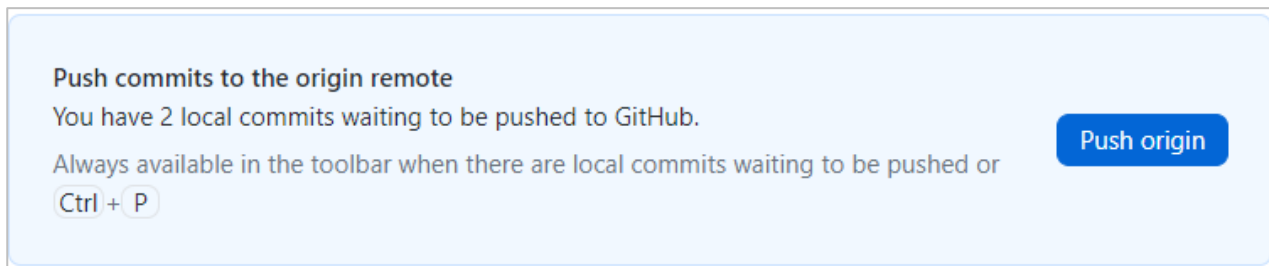
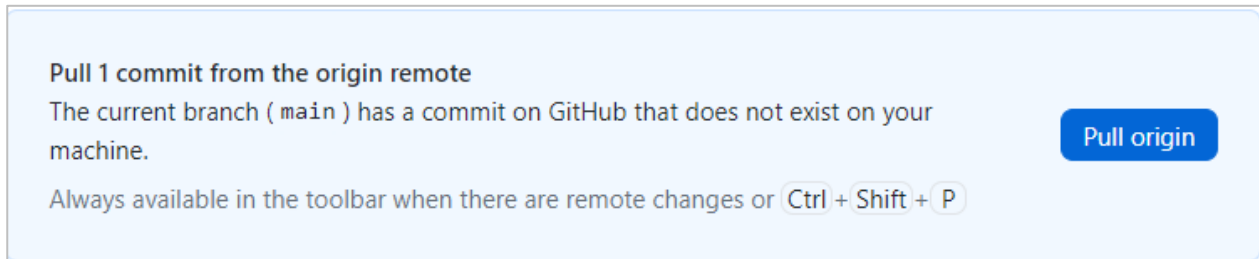
On GitHub Desktop try to commit and push the second file. A pop-up window will appear telling you that there are new commits on remote and will ask you to fetch, before trying to push. This happens because the repository in “TestRepo1” folder is older than the repository in GitHub and doesn’t contain the changes we made via GitBash.

2. Resolve the conflict

So hit Fetch button.



Then **pull** the changes made from GitBash to update and then **push** the changes from GitHub Desktop.



As you can see, we now have all commits in our repo.

