# Exercises: Git and GitHub

Problems for exercises and homework for the ["Software Technologies" course @ SoftUni](https://softuni.bg/trainings/4326/software-technologies-september-2023).

# Create a GitHub Developer Profile

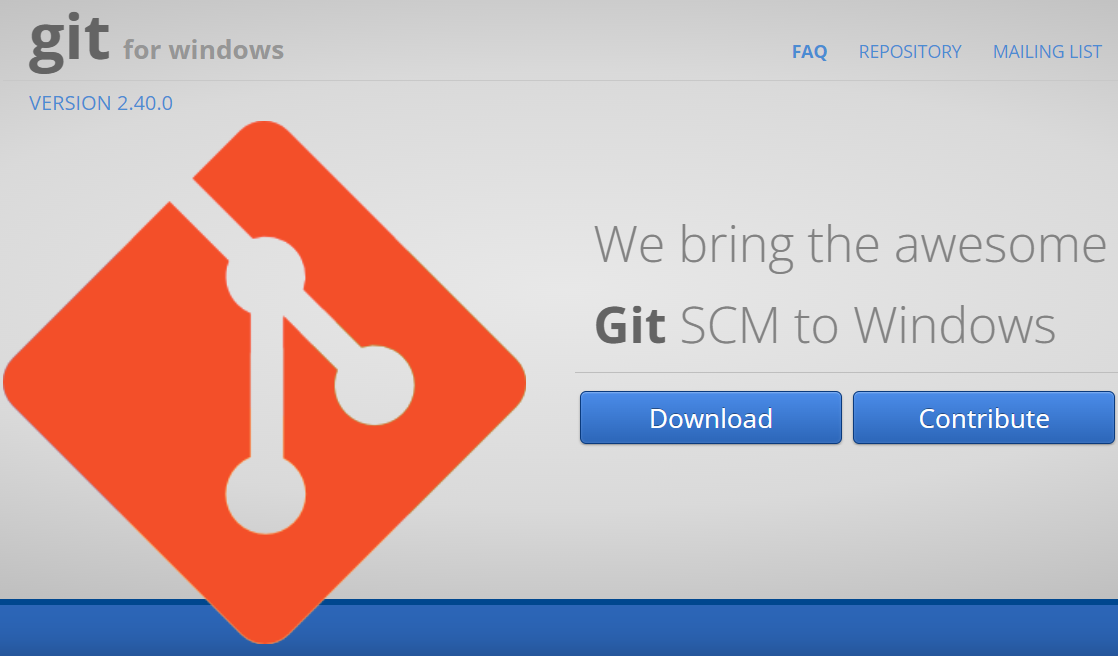
## Register a GitHub Profile

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

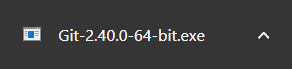


## Install Git

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



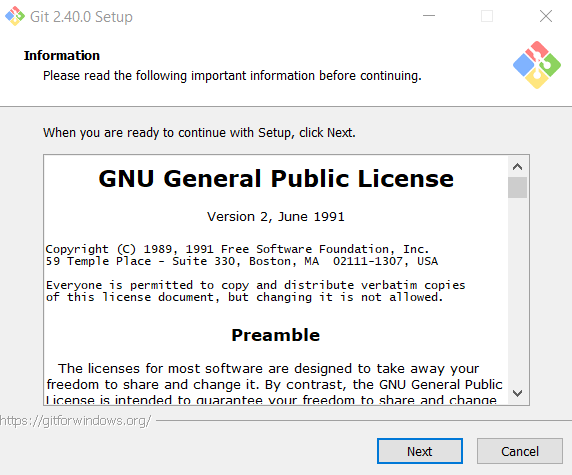
**2.** The .exe file will be downloaded to your browser.



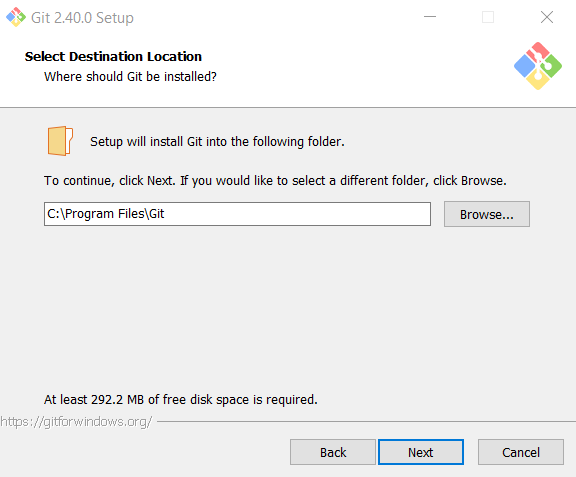
**If you need the 32-bit version, go to:**

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

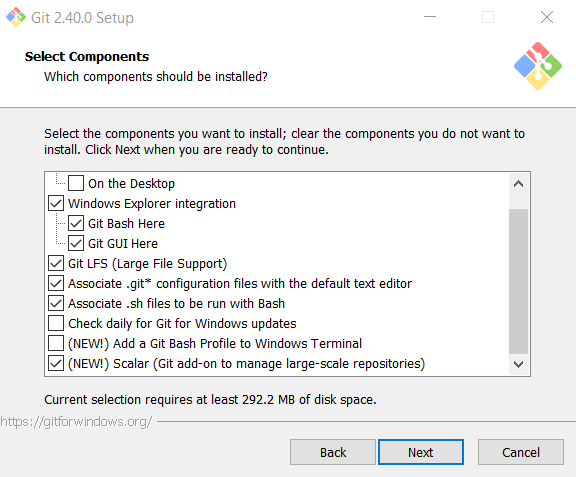
**3.** Double click on .exe file to initiate the installation process. When prompted by Windows, permit to make changes. You will get a General Public License Information screen. Click the "Next" button.



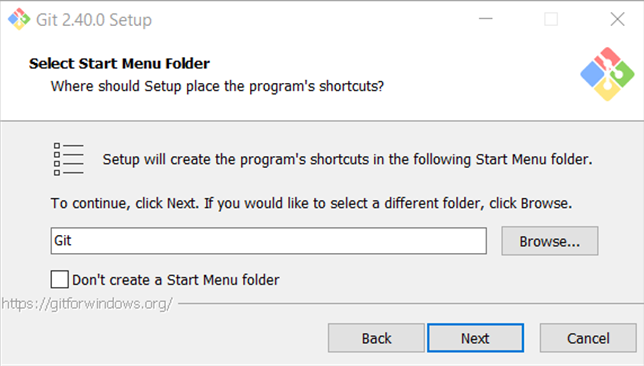
**4**. **Choose the desired path** where you want to install Git.



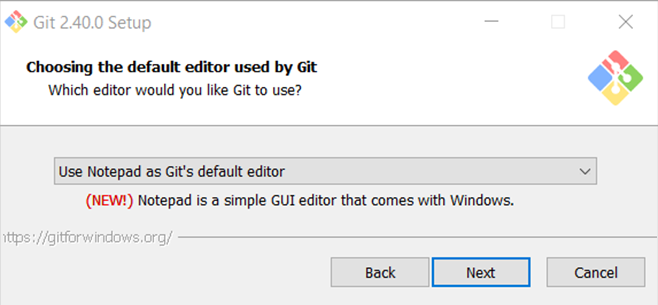
**5.** Leave the Select Components screen as it is. **Don't change anything**. Click **Next.**



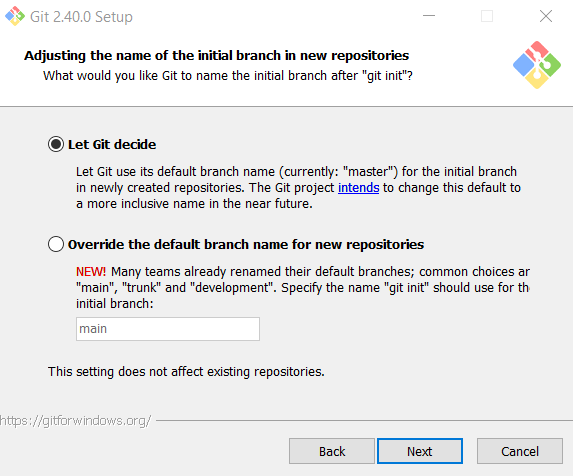
**6. Select Start Menu Folder** or not to create Start Menu Folder



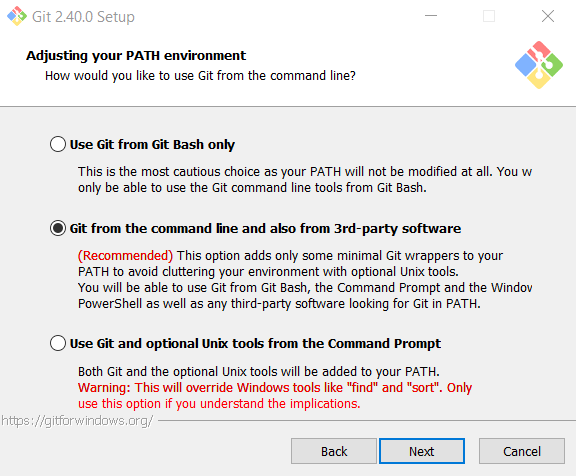
**7. Choose Notepad or Notepad++** for the default editor.



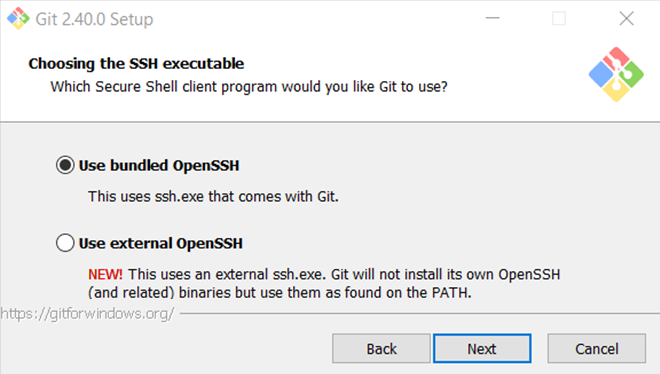
**8. Let Git decide** on the initial branch.



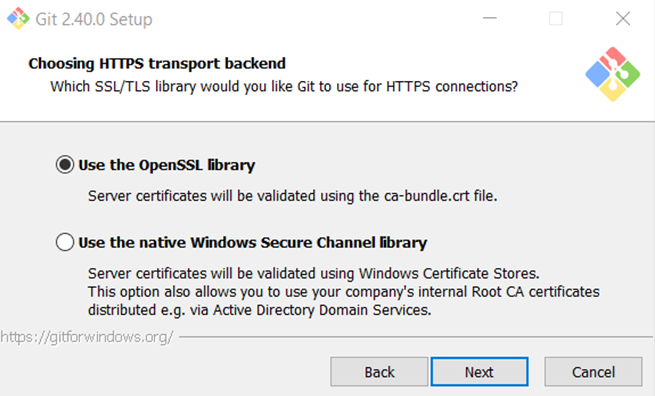
**9. Choose the second** **option** to be able to use Git from Command Prompt.



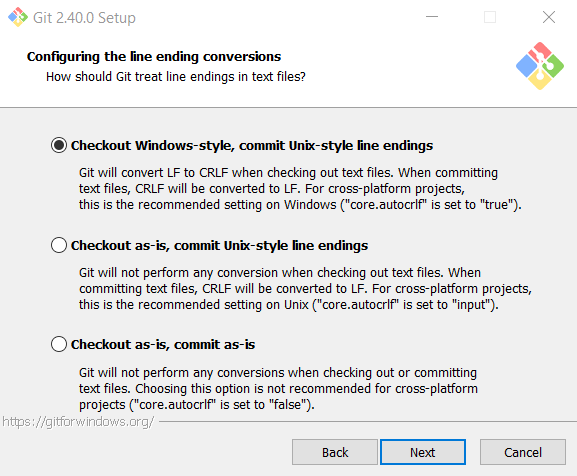
**10. Leave this option as it is.**



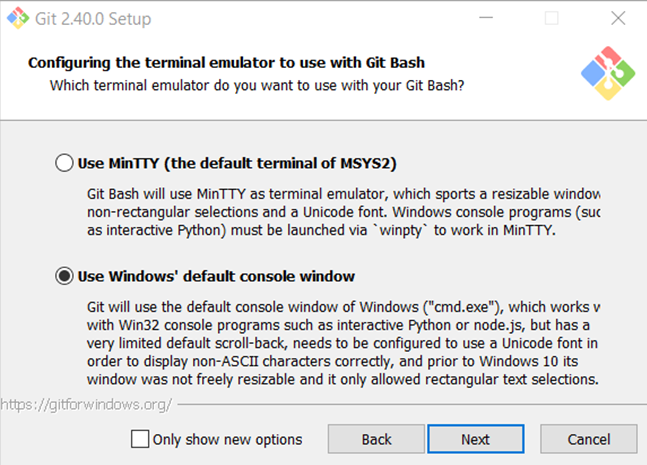
**11. Leave this option as it is** as well.



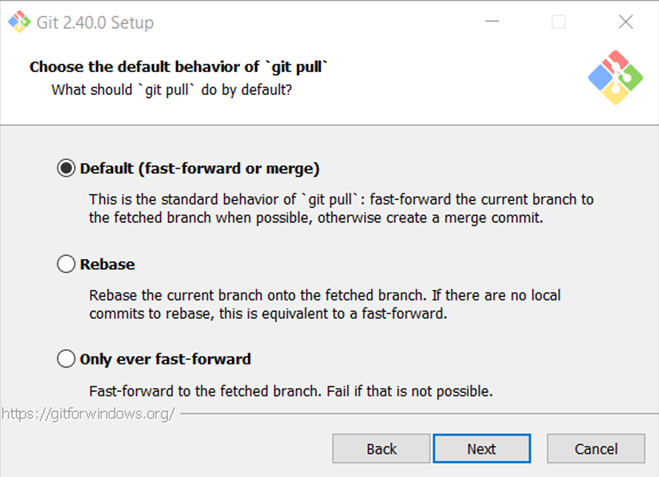
**12.** This one **stays the same** as well.



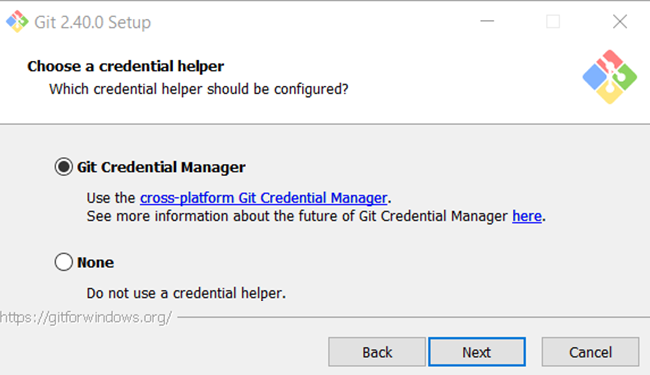
**13.** Select **Use Windows' default console window** option.



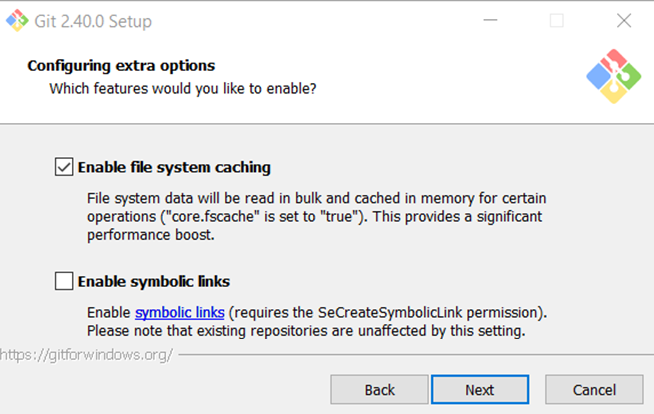
**14.** Default behavior – **Default.**



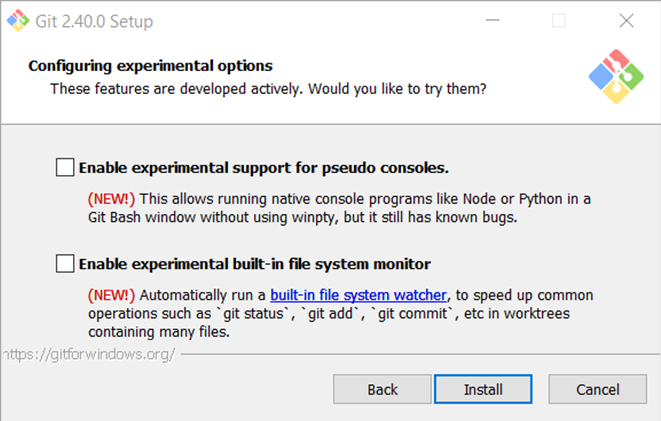
**15. Git Credential Manager**



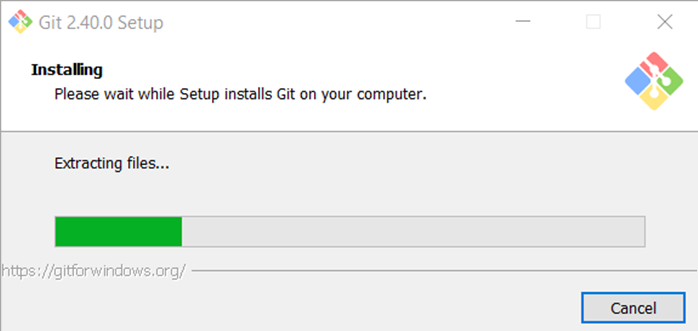
**16. Enable file system caching**



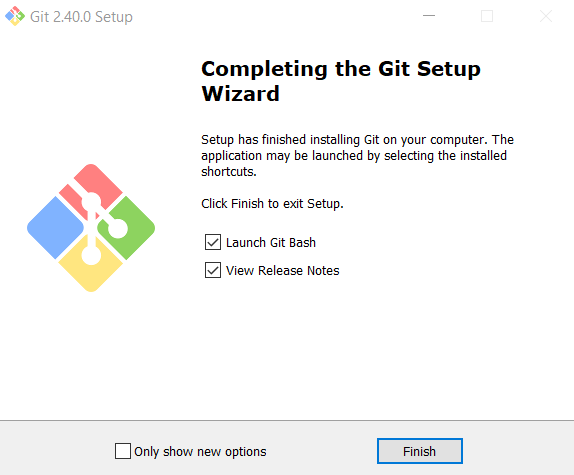
**17.** Leave the **experimental options unchecked. Click Install.**



**18.** Installing…



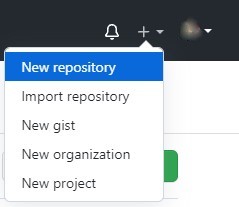
**19.** Complete the setup by pressing **Finish.**



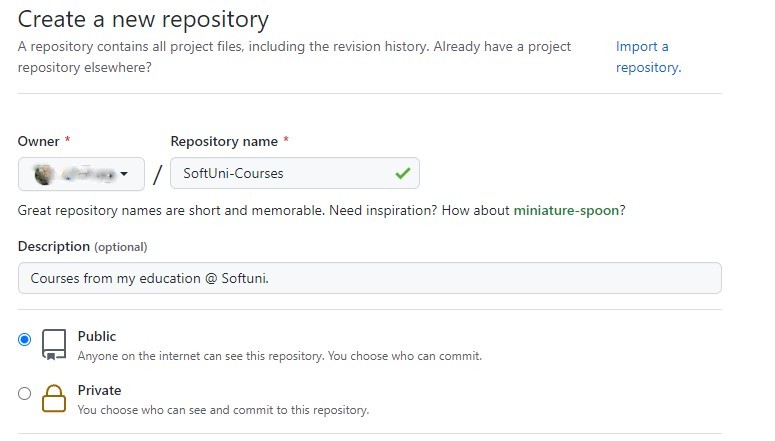
# Create a GitHub Repo and Upload Your SoftUni Projects

## Create a GitHub Repo

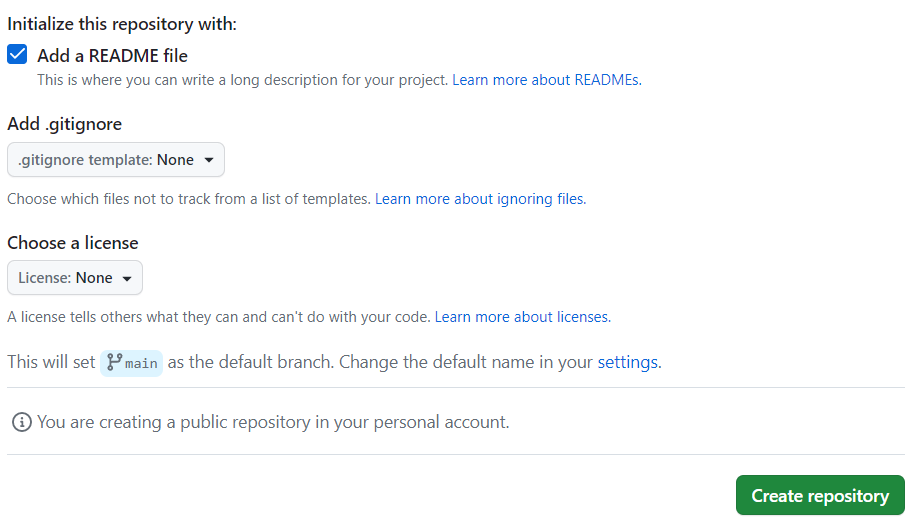
In the upper-right corner of any page, use the drop-down menu, and select **New Repository**.

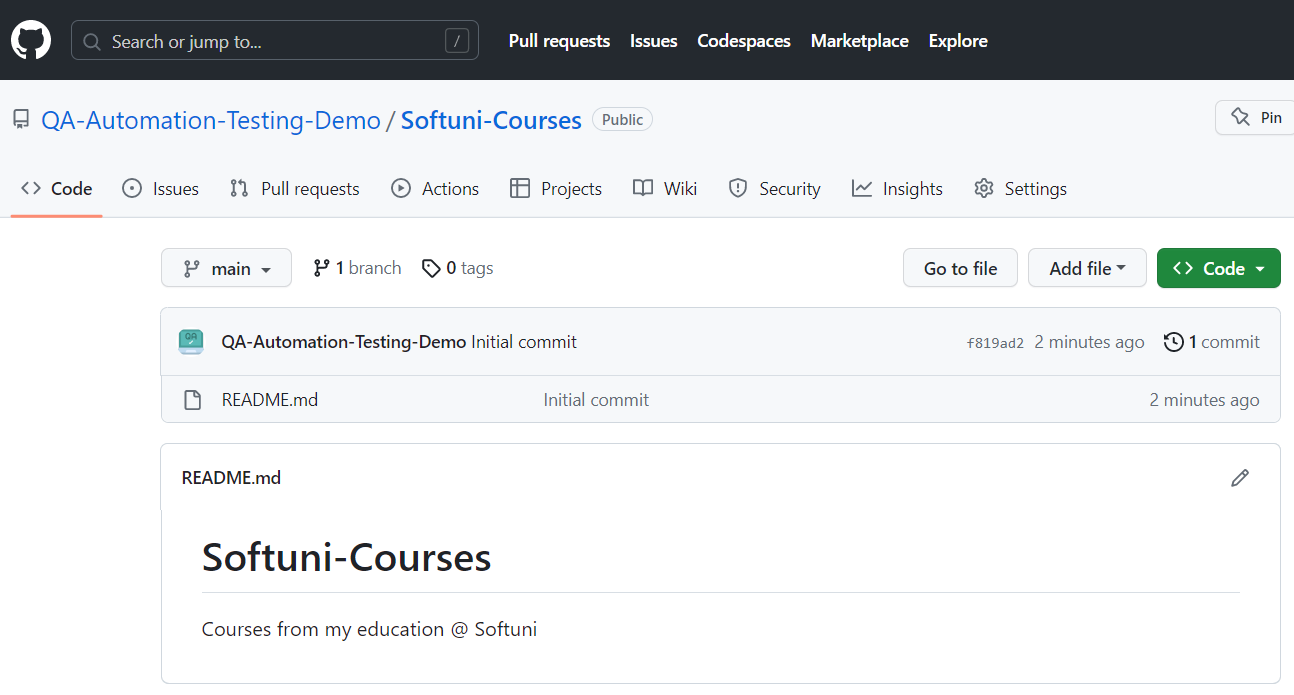


We choose a name according to the topic of our project. We can do it public or private.



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



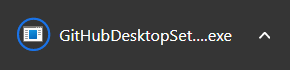
Your repository is now created. 

## Download and install GitHub Desktop

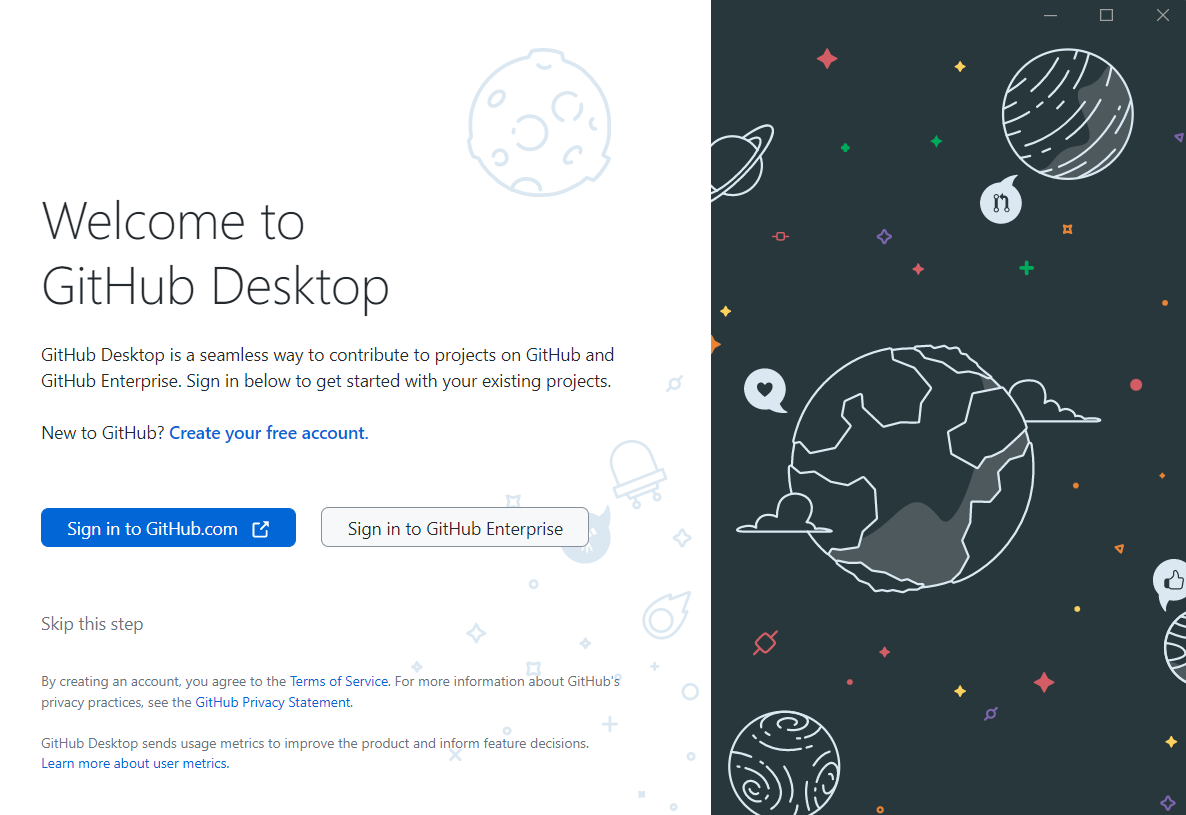
Navigate to [**https://desktop.github.com/**](https://desktop.github.com/)   
For this tutorial, we're downloading GitHub Desktop for Windows (64bit).



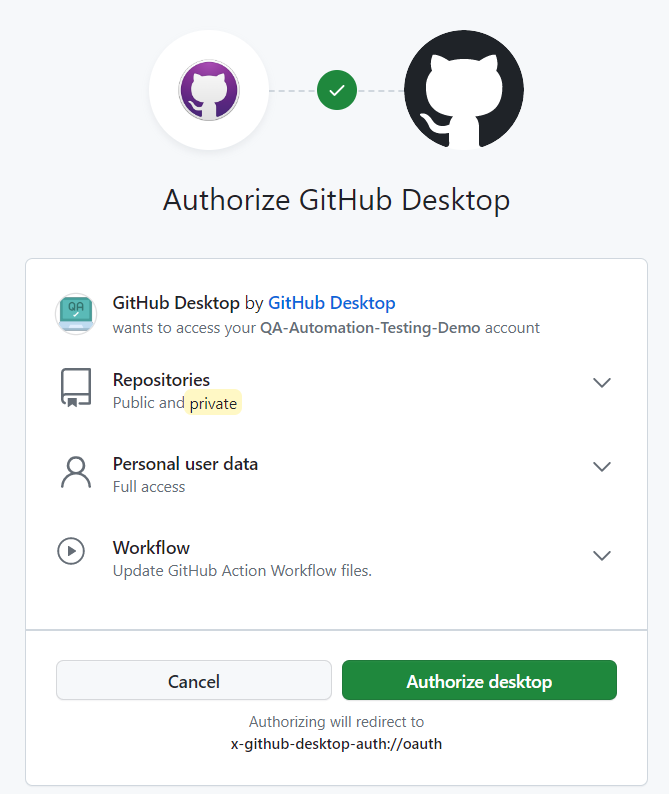
After starting the installation file…



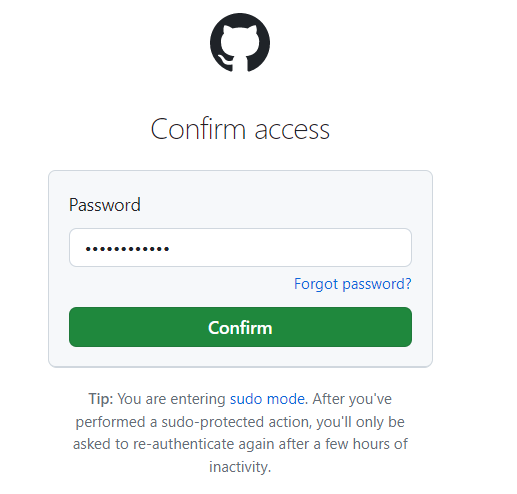
You will be greeted by the welcome screen. Since you already have an account, hit the "Sign in to GitHub.com" button



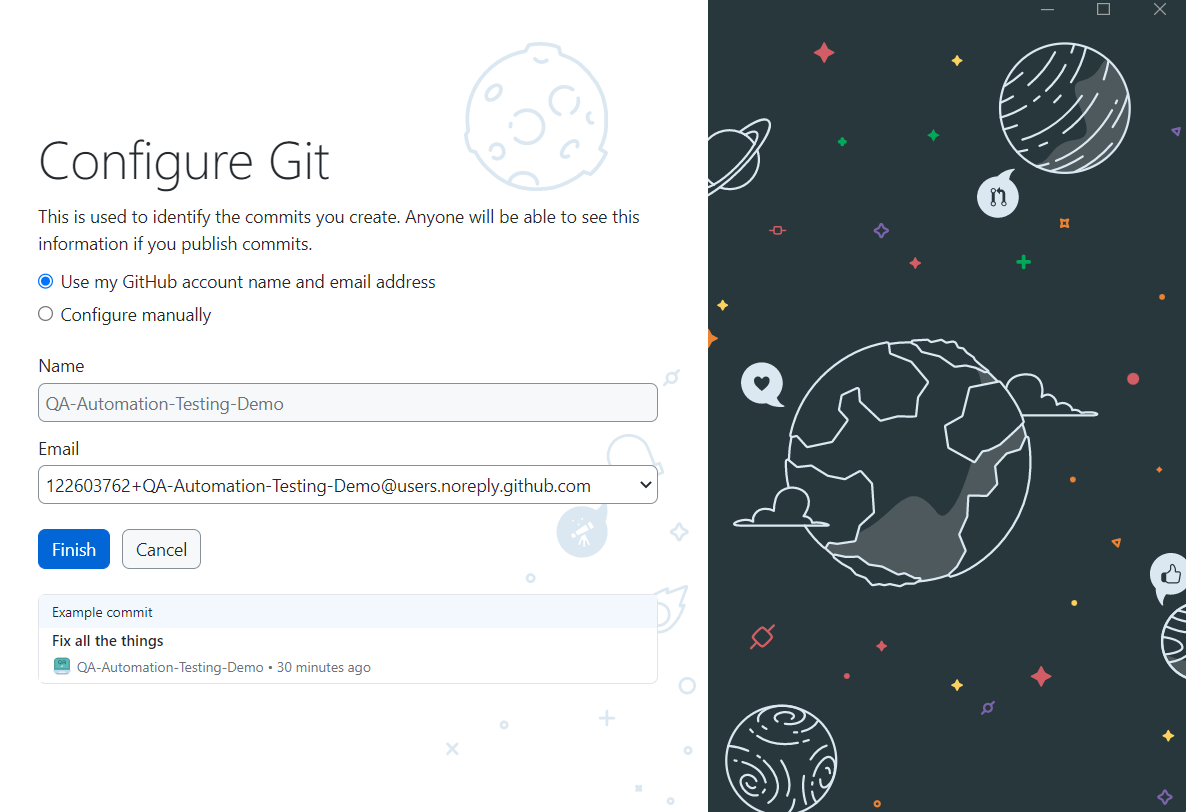
A new pop-up window will appear in your browser asking you to authorize GitHub Desktop.



And then confirm access with your GitHub password.

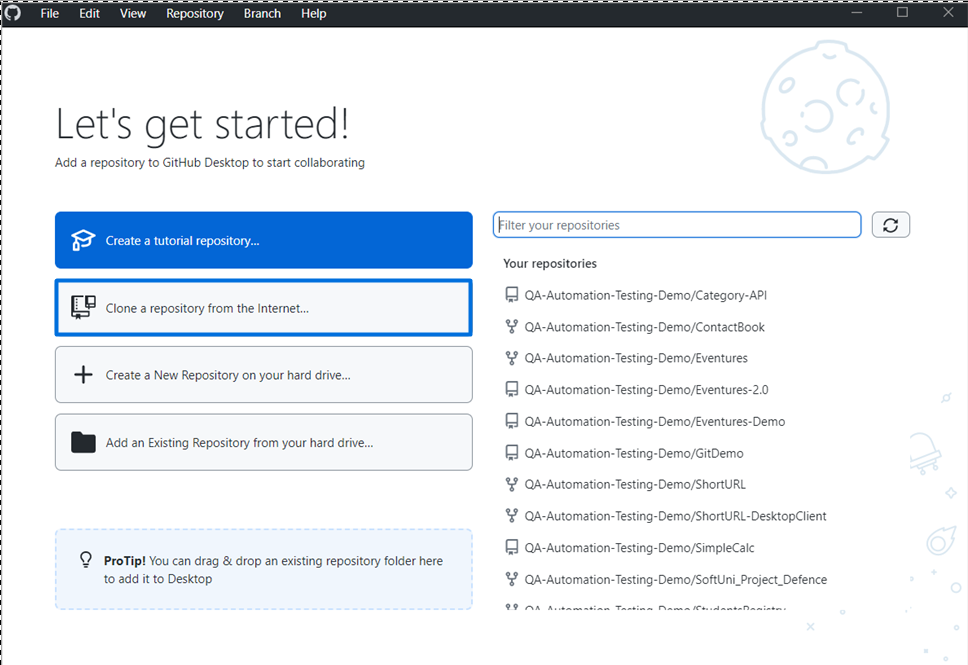


If the authorization is successful, you will be prompted to configure git. Leave it as it is and hit "Finish".

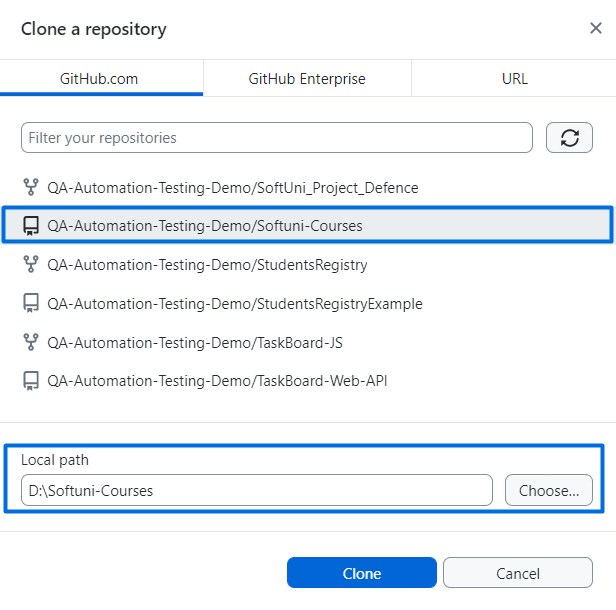


## Clone a repository

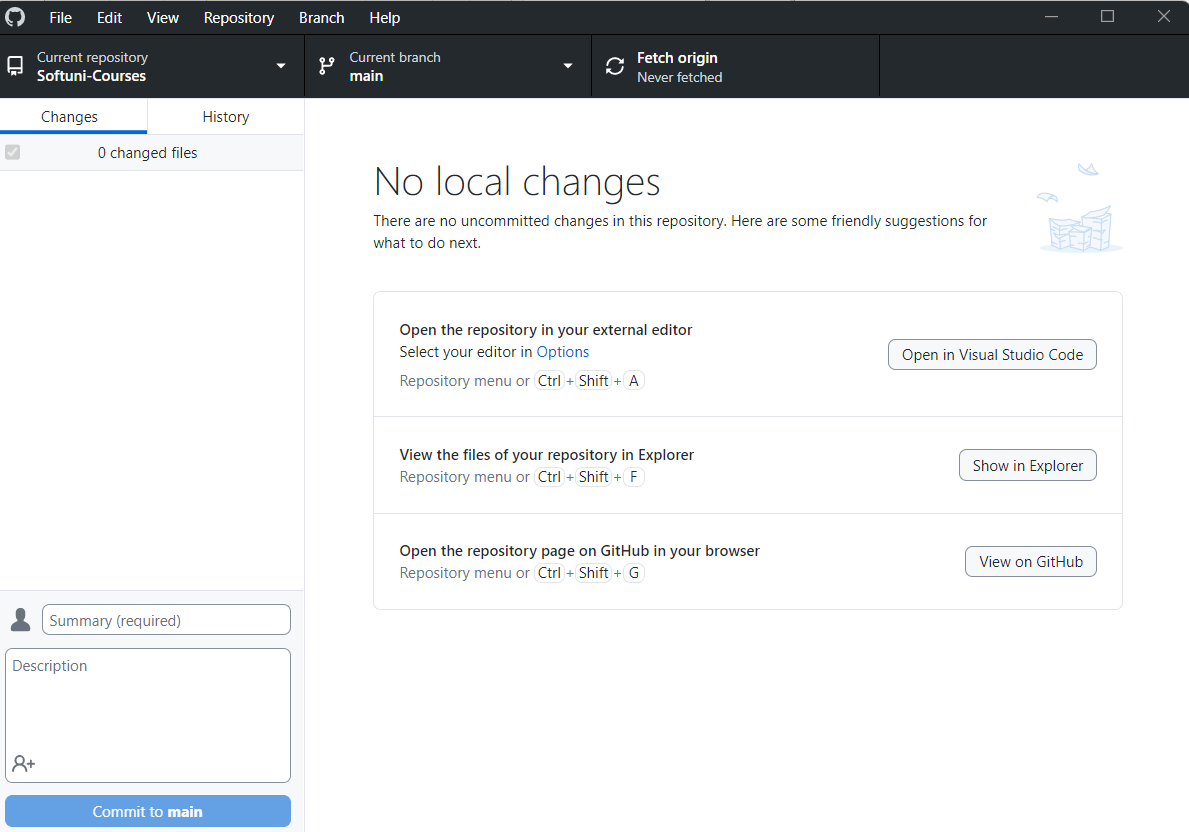
So, now your GitHub Desktop is connected to your GitHub account and you can clone the repository that you just created on your local machine. Select "Clone a repository from the Internet…".



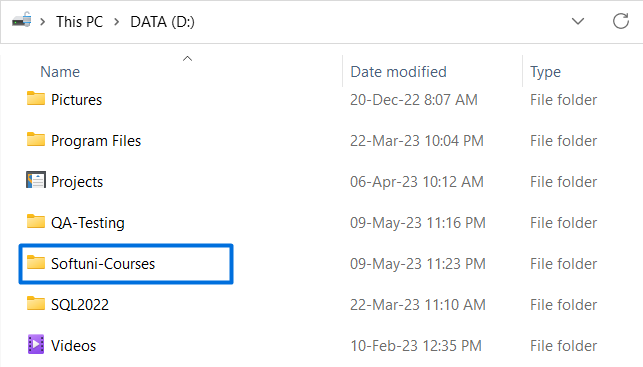
Then choose the repository that you created (The one called "Softuni-Courses"). Choose the path where you would like to save it on your local machine and click "Clone".



The repository is now cloned locally.

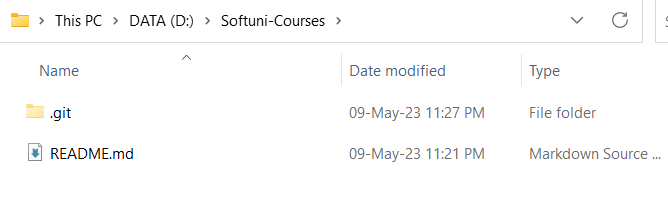


You can go and check if the new folder is properly created locally.

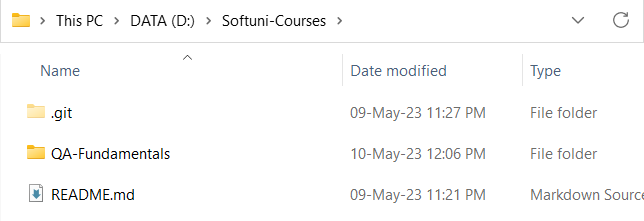


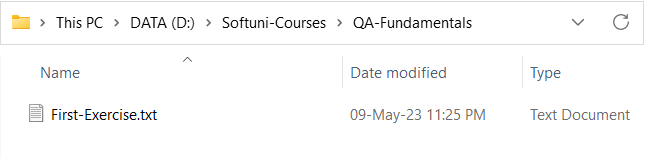
## Commit and Push

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

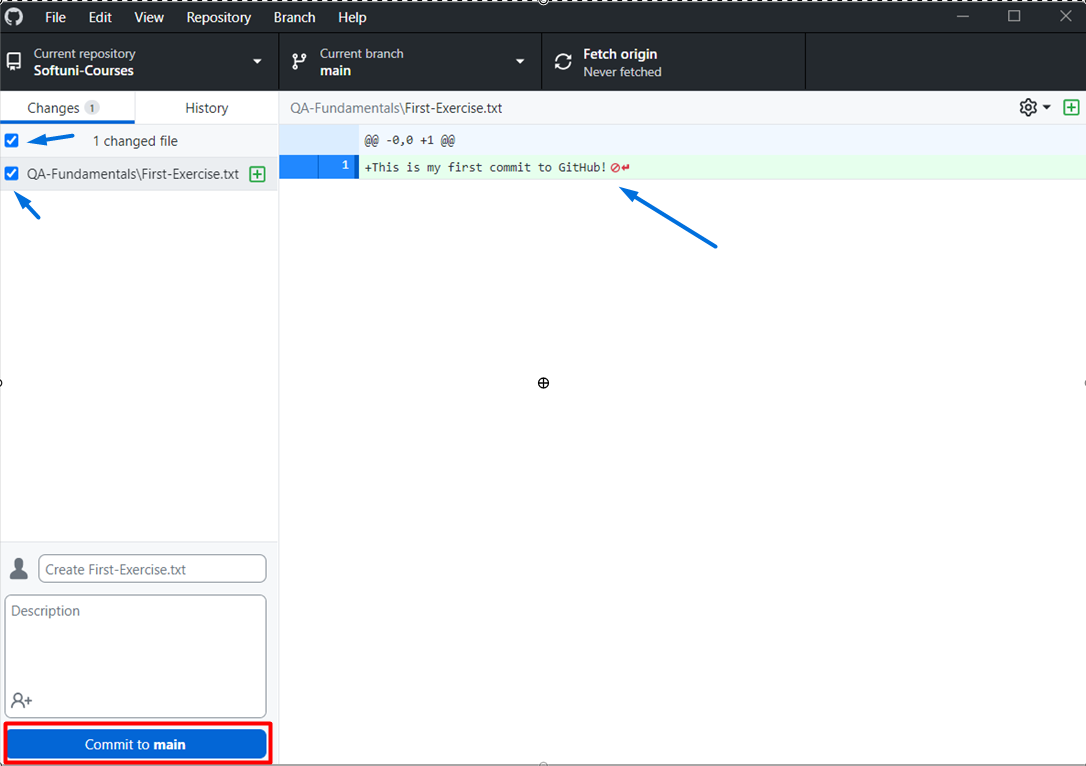


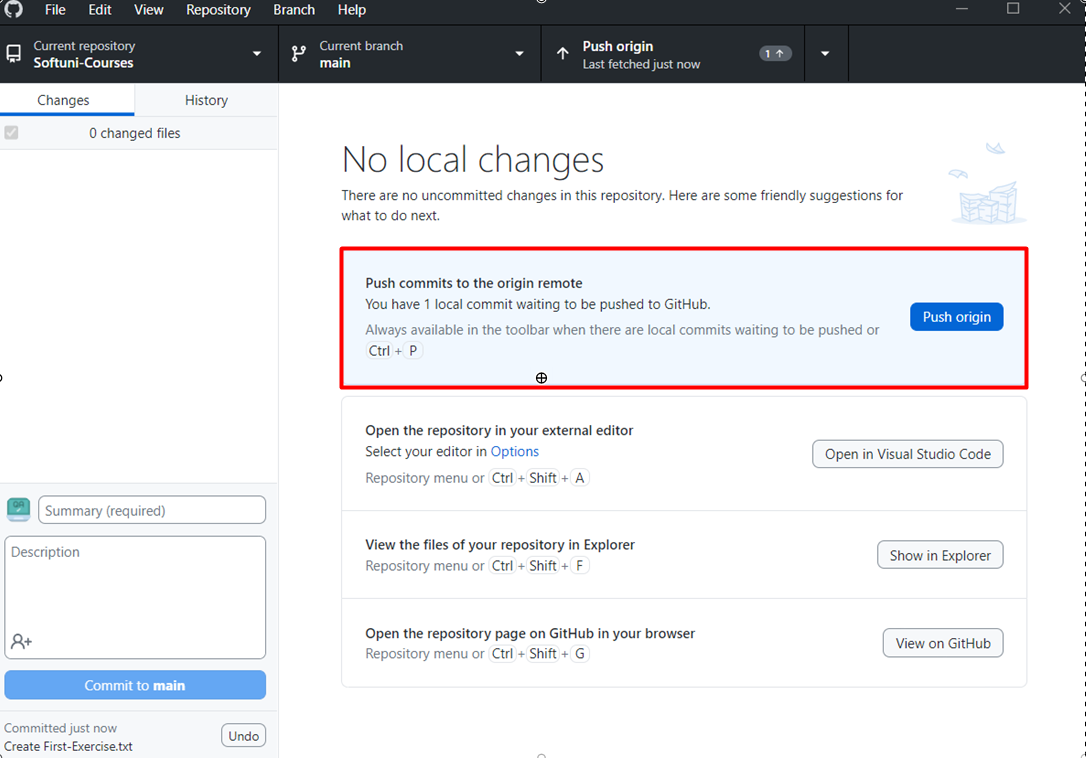
Create a new folder inside and call it "QA-Fundamentals". And inside the "QA-Fundamentals" folder, create a new text file (.txt) with an editor of your choosing and write something in it. In this case, our file is called "First-Exercise" and the text in it reads: "This is my first commit to GitHub!".



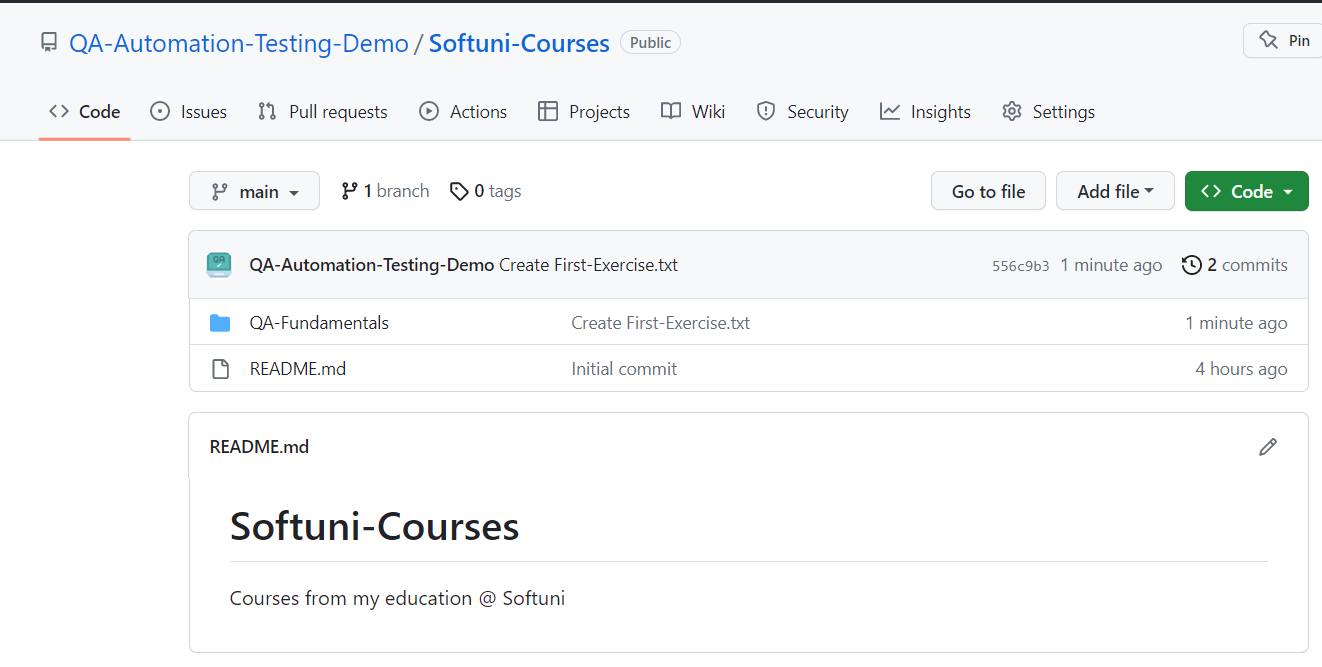


Now, head back to GitHub desktop. As you can see there is one uncommitted change that is pending. Hit the "**Commit to main**" button and then "**Push origin**".





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



# Git Bash

## Check Git version

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

* from the command prompt **git version**
* from git bash **$ git –version**

## 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.

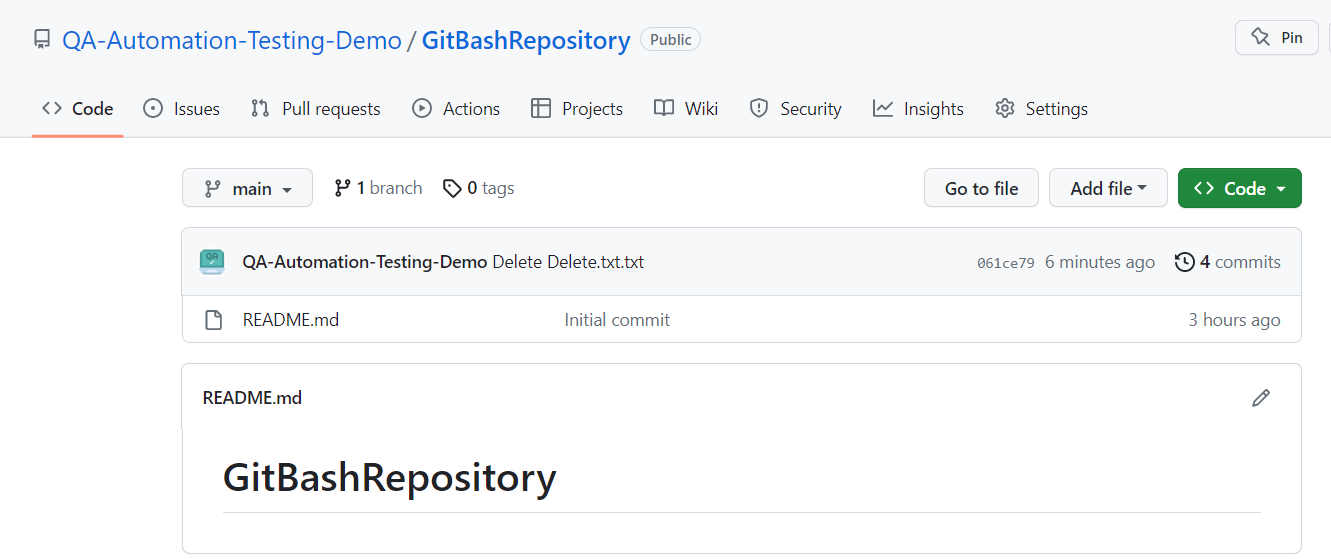
**If you don't see any output** or **the output does not include the user.name and user.email settings**, it means that Git is **either not installed or not configured on your machine**. In that case, you can follow the installation instructions for your operating system and **then configure Git by running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

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

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

## Clone GitHub repository

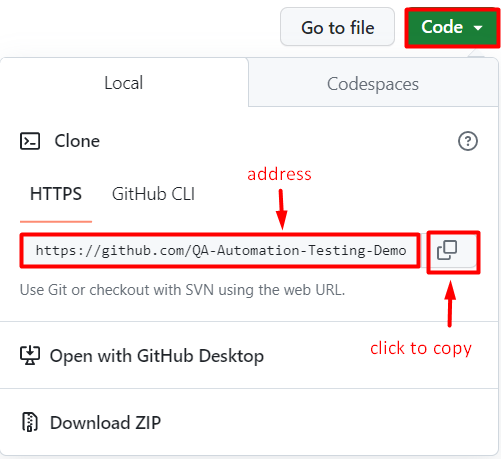
Since you already know how to create repositories on GitHub, go and create a new one.



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:

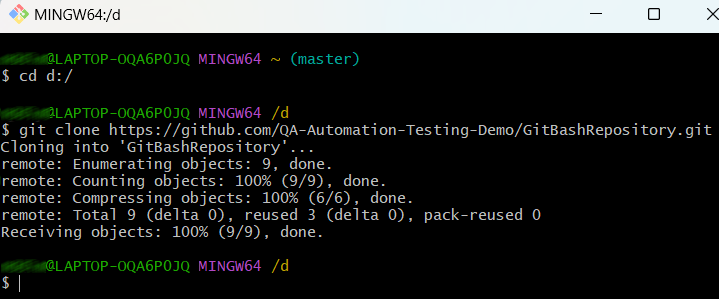
**$ cd d:/**

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


Type the following command:

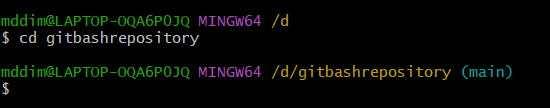
**$ git clone <address-of-your-repo>**

This is how it should look like:

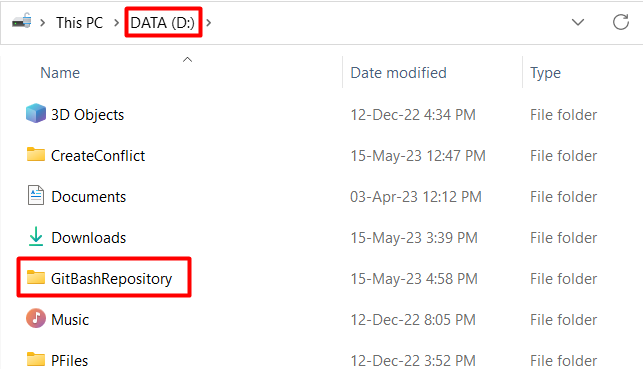


Navigate to the newly created directory by running the following command:

**cd <repository-name>**

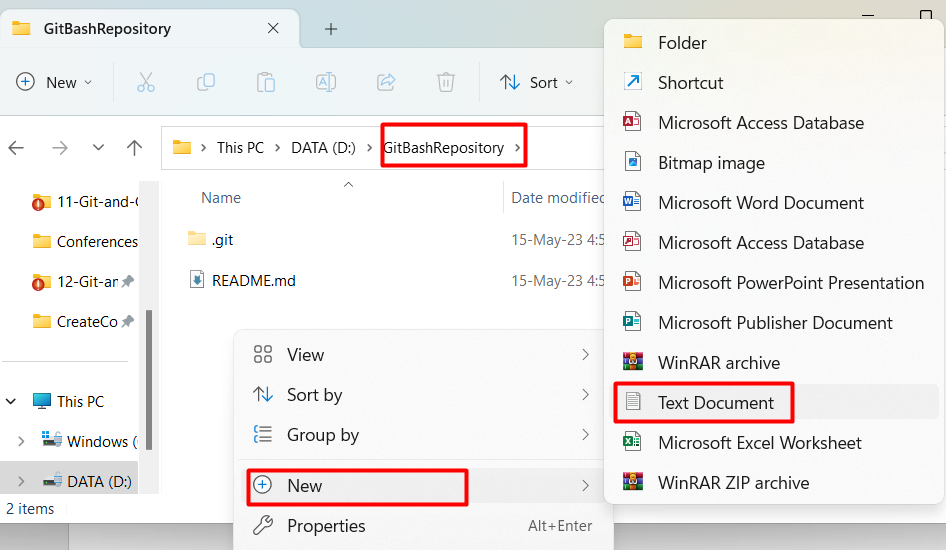
****

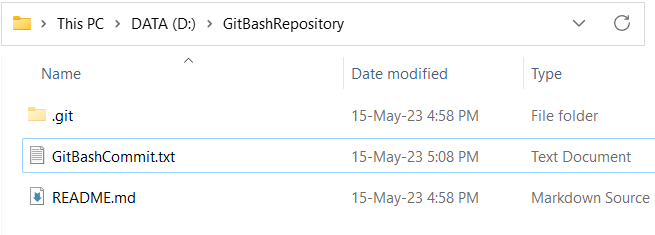
You can also check via Windows Explorer if everything is cloned on your machine.



## Edit, Commit, and Push

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





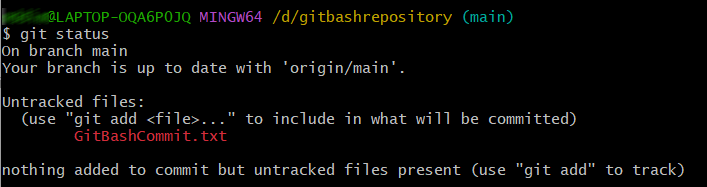
Use your favorite text editor or IDE to modify files in the repository.

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**

****

Let's commit your changes to GitHub.

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

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

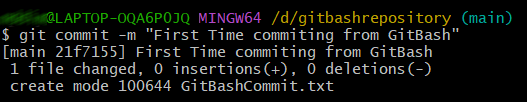
Alternatively, to stage all changes, you can use:

**$ git add .**

****

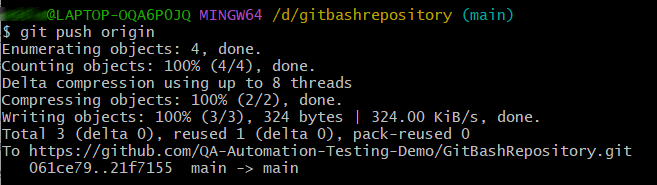
Commit the staged changes with a descriptive message:

**$ git commit -m "Commit message"**

****

Push your commits to the remote repository:

**$ git push origin**

****

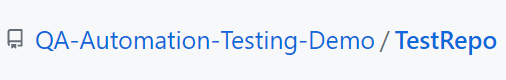
Congratulations! You have successfully created a repository, cloned it, made changes, committed them, and pushed them back using GitBash.

# 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.

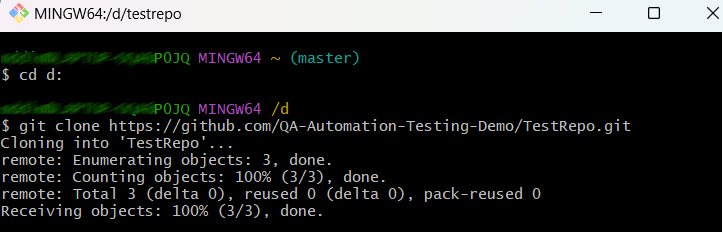
## 1. Create a conflict

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



**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.



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**

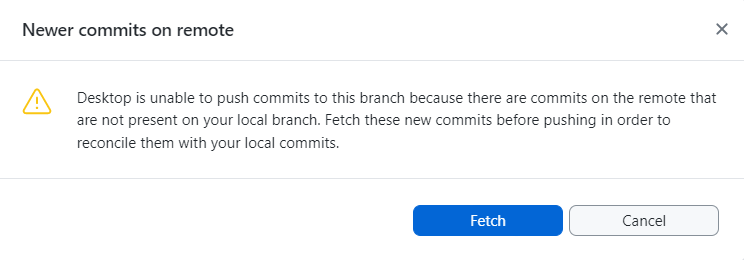
**On GitHub:** Go and check if the file that you just pushed is uploaded.



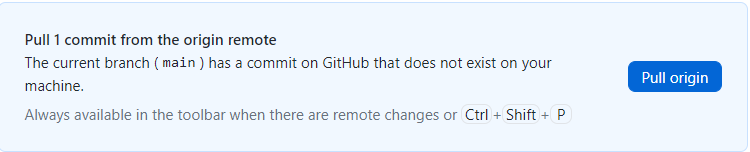
Now, in "TestRepo1" folder create another .txt file, give it a name add some text inside and save the file. For example, we named our file **CommitFromGitHubDesktop.txt.**

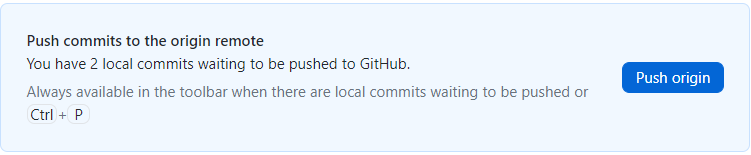
**On GitHub Desktop** try to commit and push the second file. A pop-up window will appear telling you that there are new commits on remote and will ask you to fetch, before trying to push. This happens because the repository in “TestRepo1” folder is older than the repository in GitHub and doesn’t contain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

## 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.  
