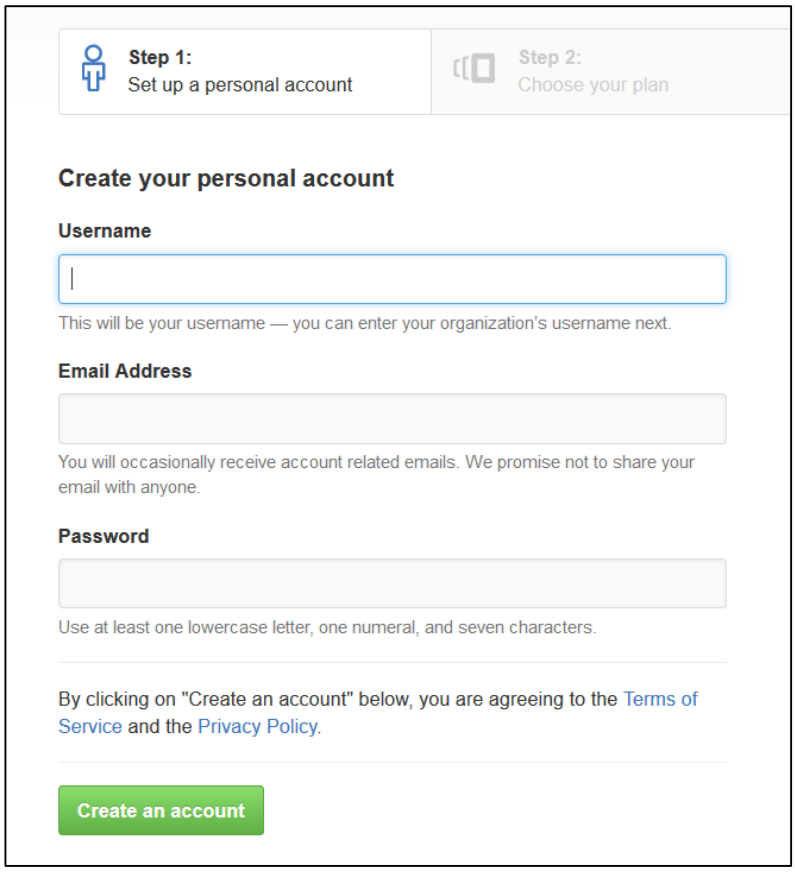
Lab: Git and GitHub

Problems for exercises and homework for the "Programming Fundamentals" course @ SoftUni.

I. Create a GitHub Developer Profile

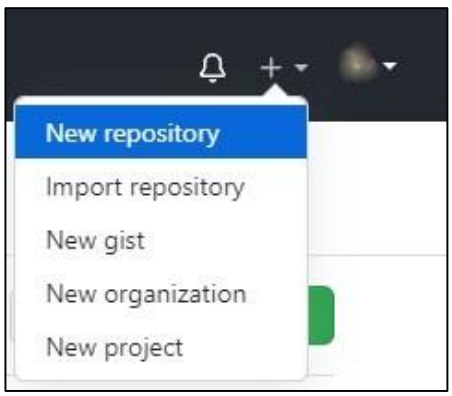
1. Register a GitHub Profile

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

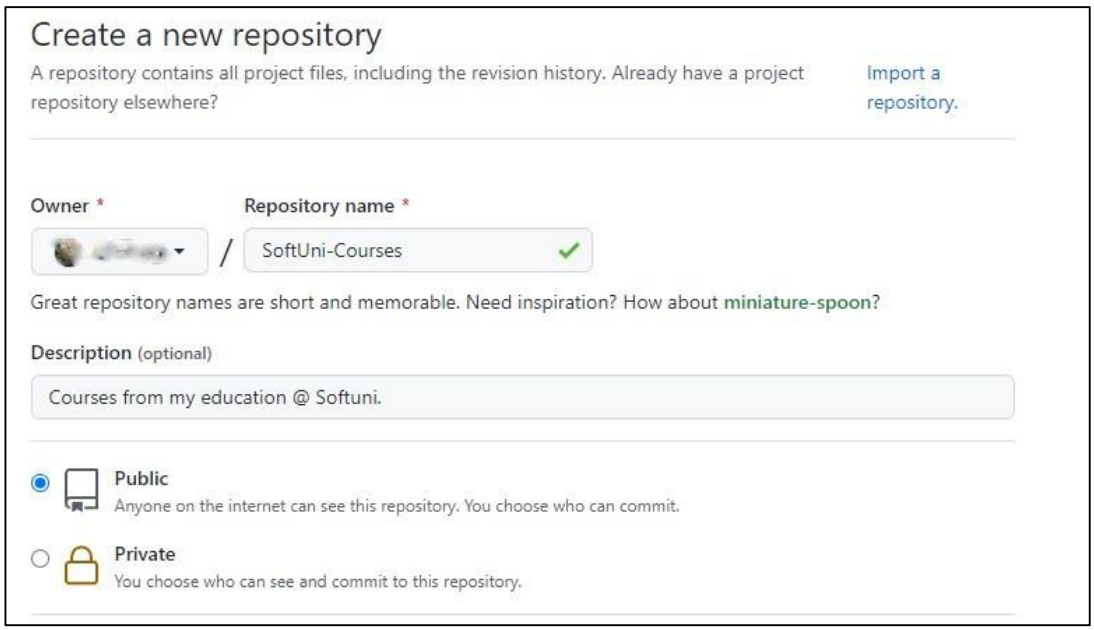


II. Create a GitHub Repo and Upload Your SoftUni Projects

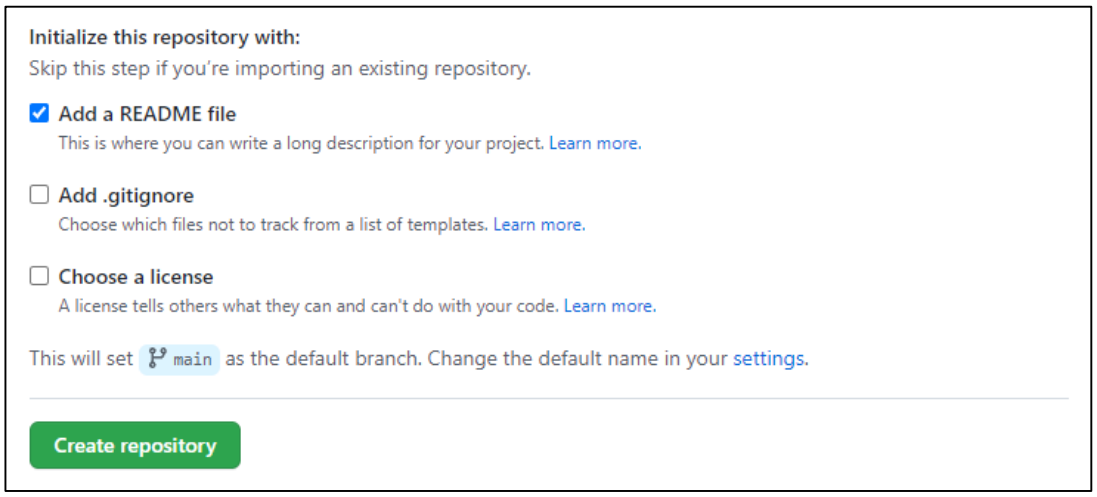
In the upper-right corner of any page, use the drop-down menu, and select **New repository**.



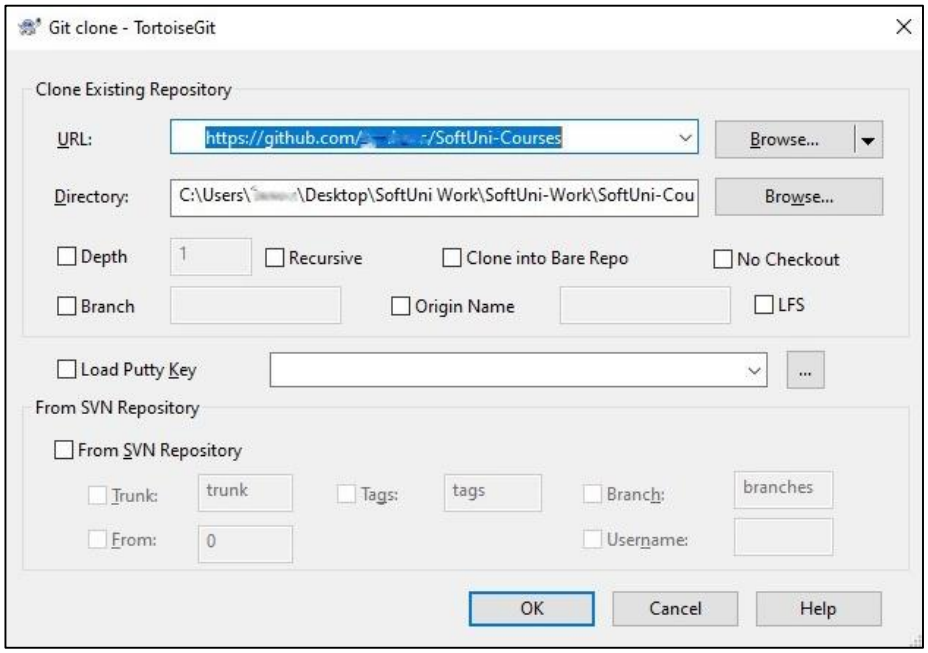
We choose a name according to the topic of our project. We can do it public or private.



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

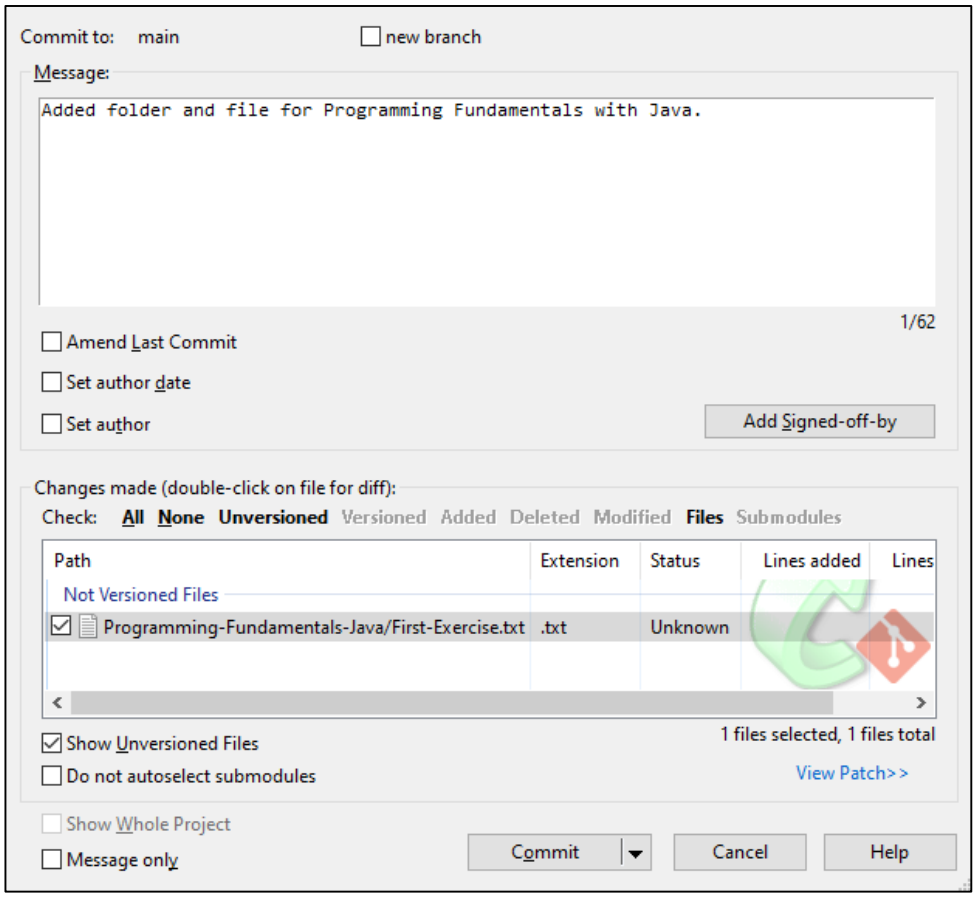


Then we select the folder on the computer in which we want to download the repo. **Right button** in folder **-> Git Clone**.

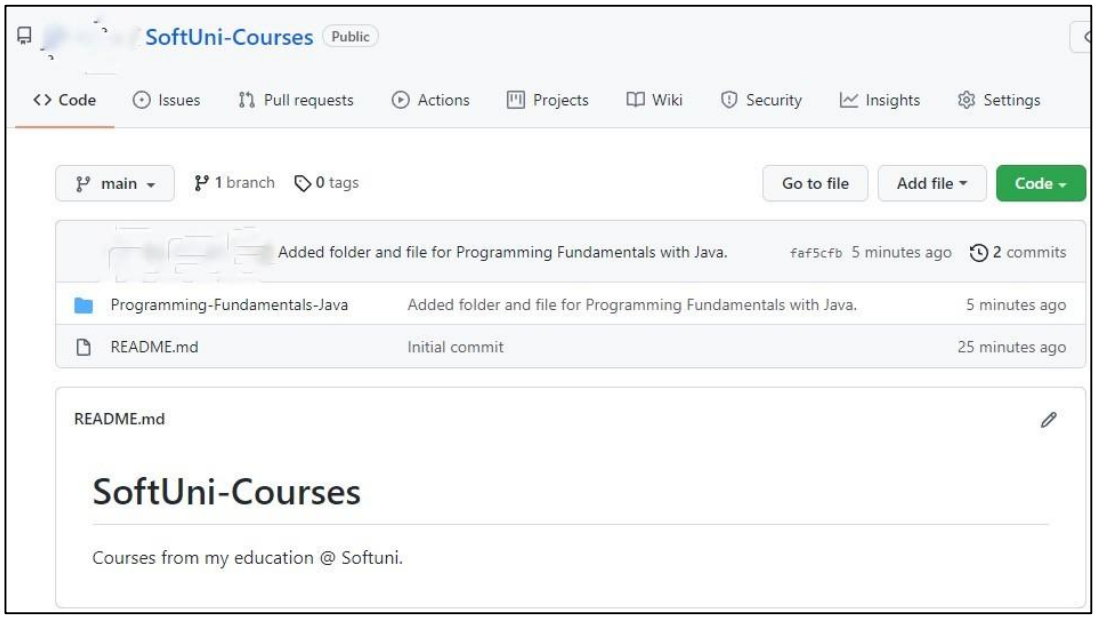


And click on **OK**.

For example, in the folder we downloaded from GitHub (SoftUni Courses), we **created** a folder with the name **Programming-Fundamentals-Java**. In it, we **created** a file with the name **First-Exercise**. We return to the folder SoftUni Courses, and right button in folder -> Git commit -> "main".



And click on **Commit**. Then in the lower-left corner, **click** on **Push.** We **return to GitHub** and see that the changes are **reflected**.

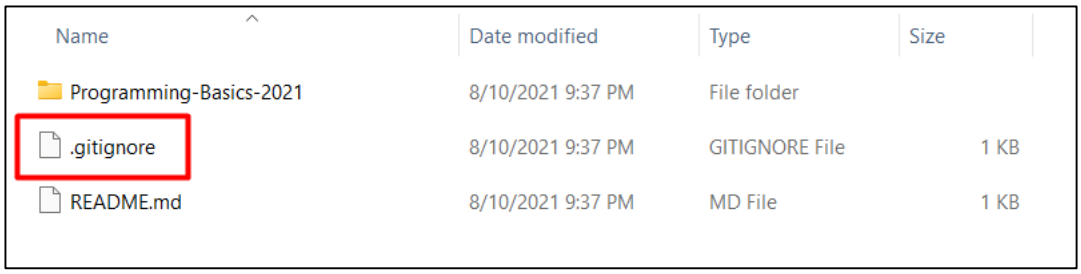


We are ready!

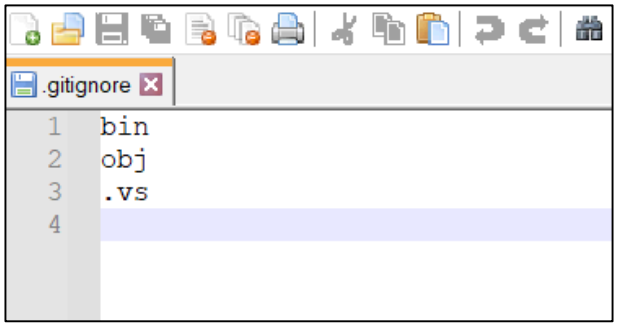
III. Configure the .gitignore file

A **.gitignore** file specifies **intentionally untracked files** that Git should ignore. These are all the files that are generated during the project build and compilation. Your repo should keep the source code + documentation + project resources but should ignore all files built from the source code.

Create a file called **.gitignore** in your project's directory:



Each line in the **.gitignore** file specifies a pattern to ignore. For example, if you code in C#, you can use these ignore settings:



If you are unsure what files you should ignore, follow this tool: https://www.toptal.com/developers/gitignore.

IV. Conflict + Resolve

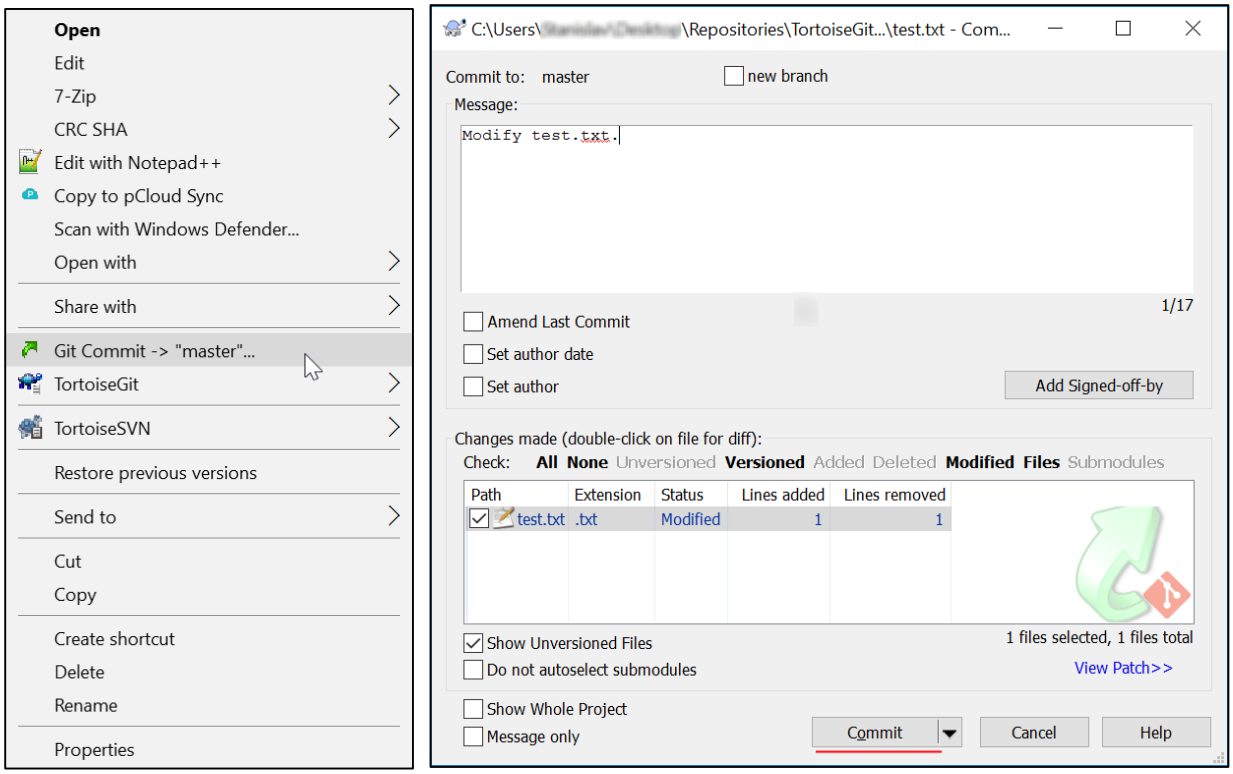
1. Make a Conflict

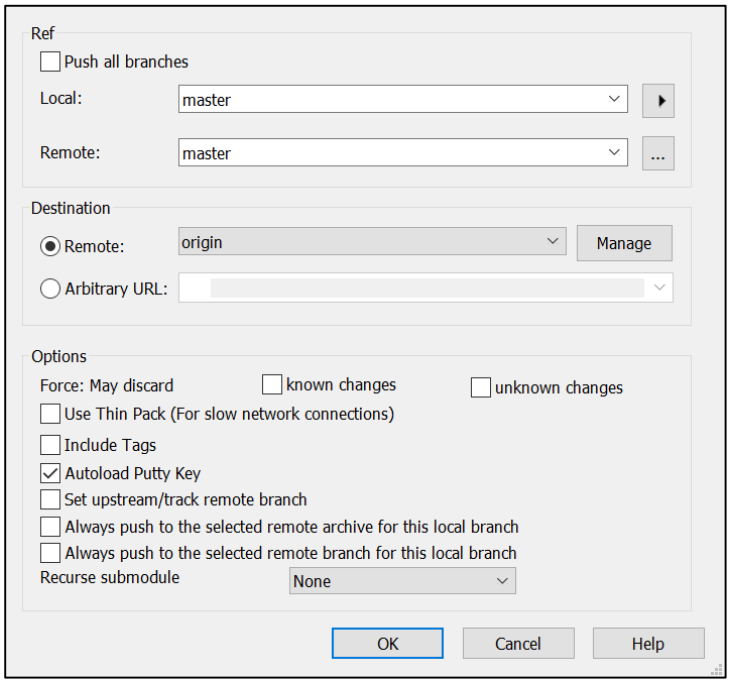
Update the content in both directories separately:

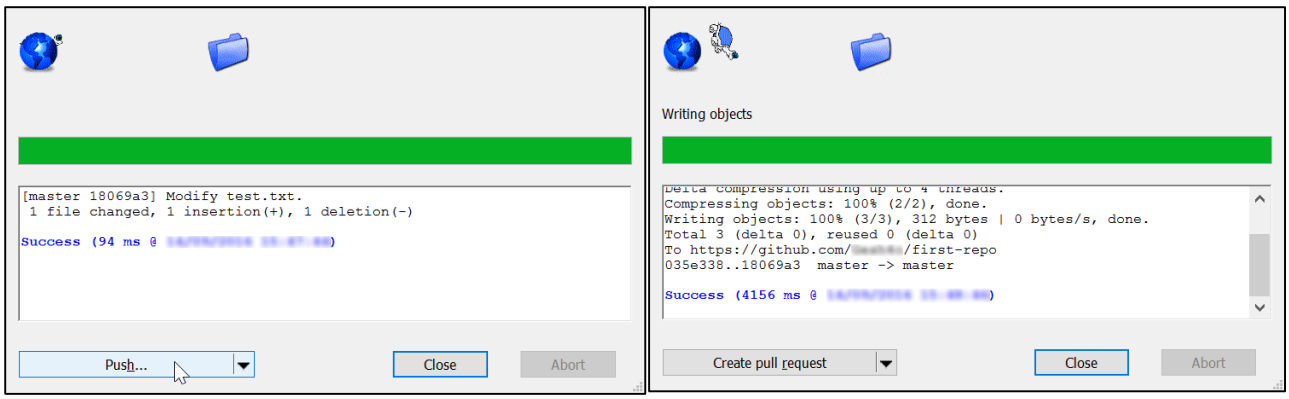
- On your **TortoiseGit** clone, create a "**test.txt**" file and add the line: "**Creating with Tortoise…**"
- On your **GitBash** clone, create a "**test.txt**" file and add the line: "**Creating with Bash…**"

2. Upload Your Changes: Commit and Push

Commit and **push** your changes from the **TortoiseGit** Clone to GitHub. You can use TortoiseGit's "**Git Commit…**" and "**Git Commit…**" commands:

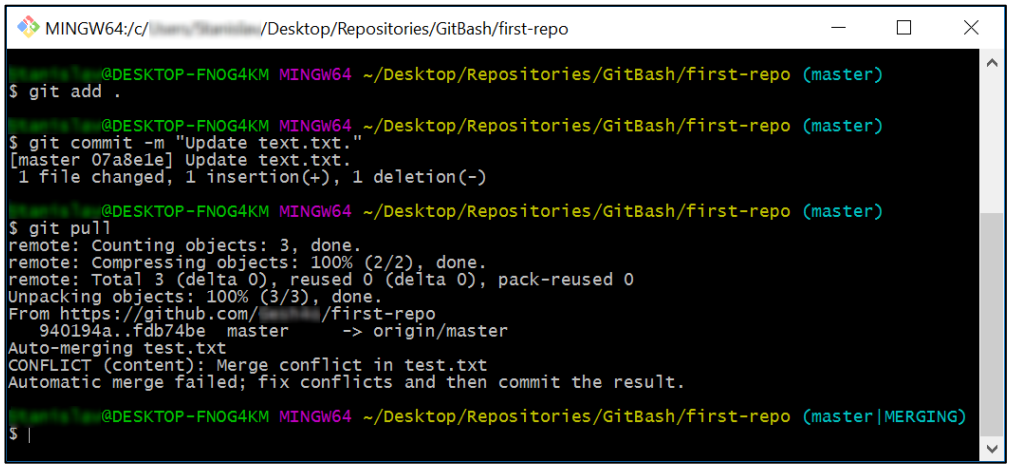






3. Update Your Bash Clone

- Open your Git clone directory and open the **GitBash** console. Run the following commands:
  - Add all modified files to the Git local **staging** area
    - "**git add .**"
  - **Commit** your changes, and give a meaningful **commit message**.
    - "**git commit -m "Update test.txt.**""
  - **Update** your local repository (get the latest changes from GitHub)
    - "**git pull**"

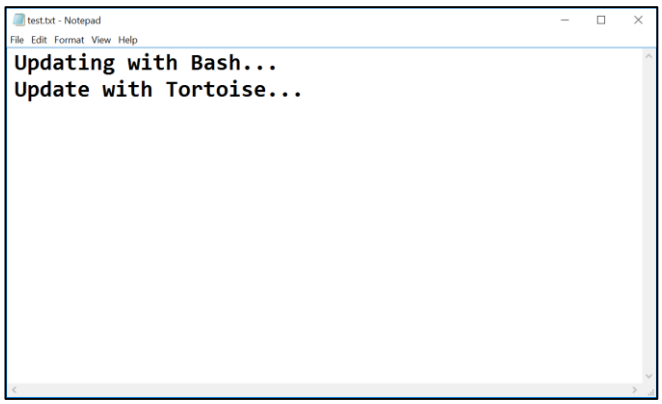


4. Merge a Conflict

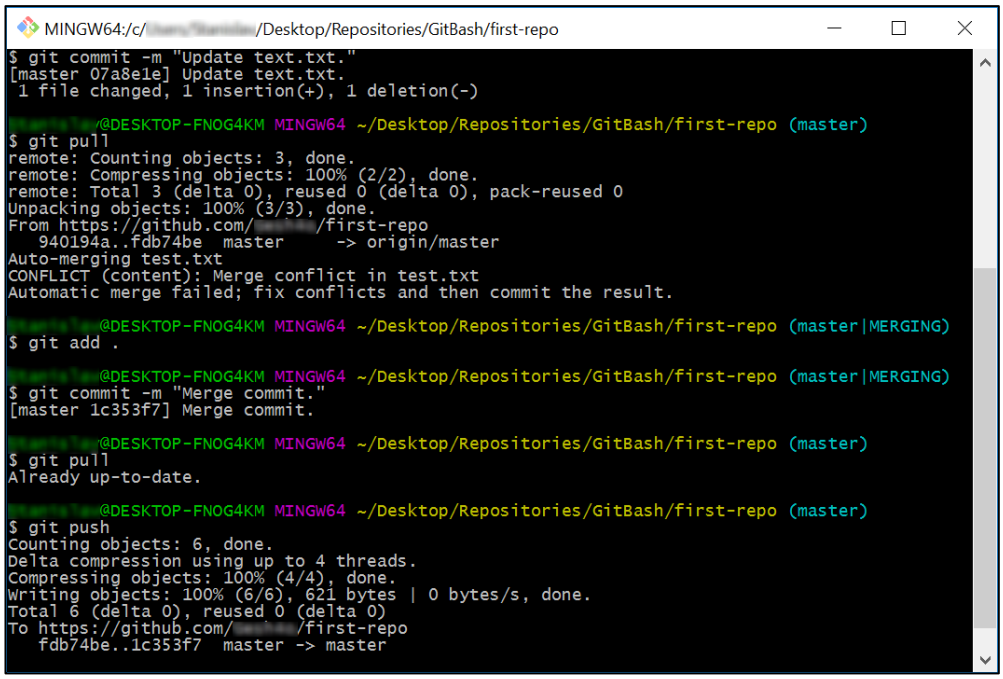
Now you have a "**merge conflict**" which you have to resolve. The **"git pull"** command **automatically** created it because the remote repository at GitHub had a newer version for some files of your code.

- **Open** the "**test.txt**" file in your **GitBash** clone. It should look like this:

- Remove the HEAD, ======, <<<<<<, >>>>>>> symbols and save the file.



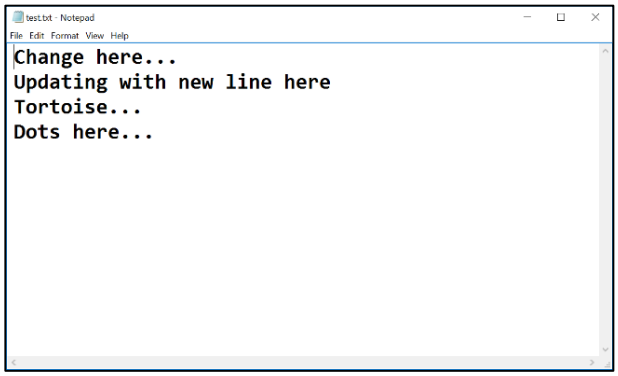
- Now that you have resolved the **conflict - stage** the modified file, **commit** again and **sync** with the remote repository (**pull** + **push**).



5. Merge Changes and Push to GitHub

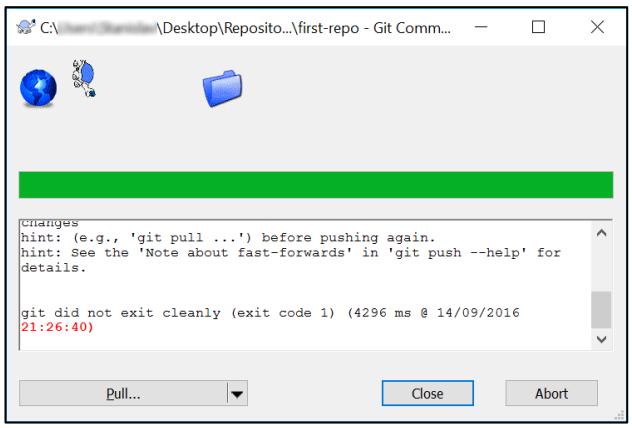
You have updated the content of your remote repository. Now try to **update** your TortoiseGit clone.

- Make additional changes to the file **test.txt** and **commit** them.

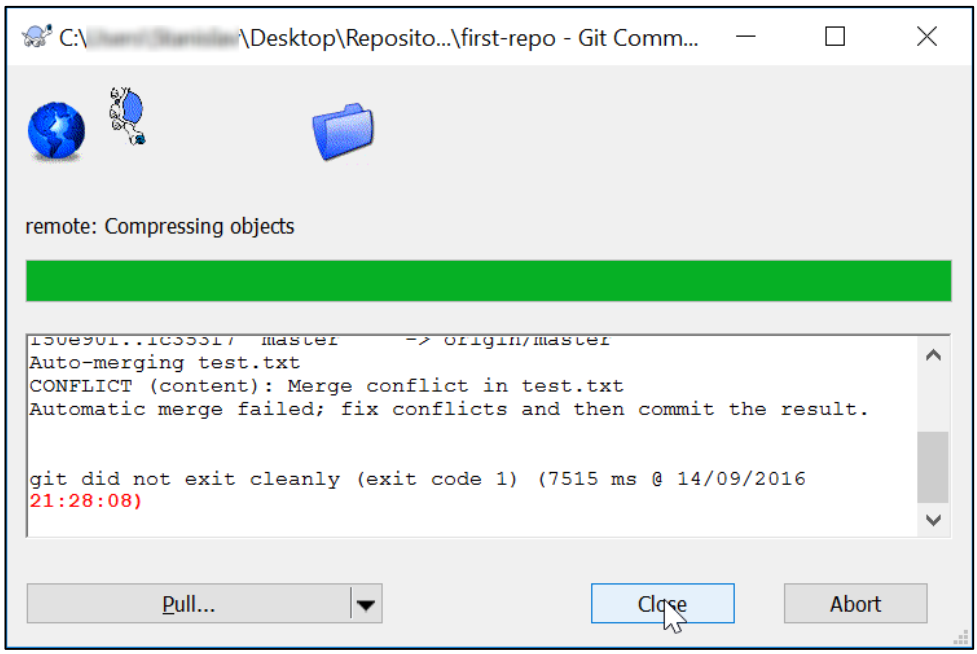


***Note** that if your changes are simple (e.g. just a new file is added), TortoiseGit may **automatically** merge them.

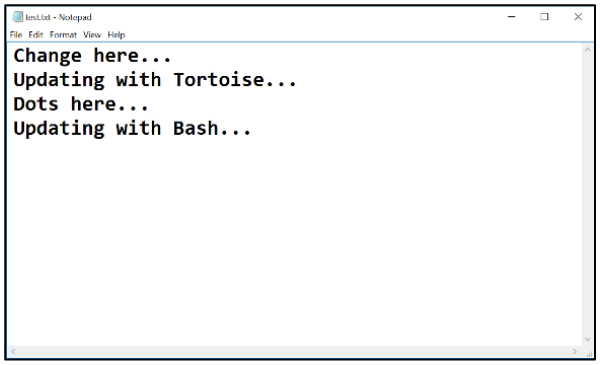
- Now try to **push**. It turns out that we have our **remote** repository **updated** (the merge commit), and you do not have these changes on our **local** repository.

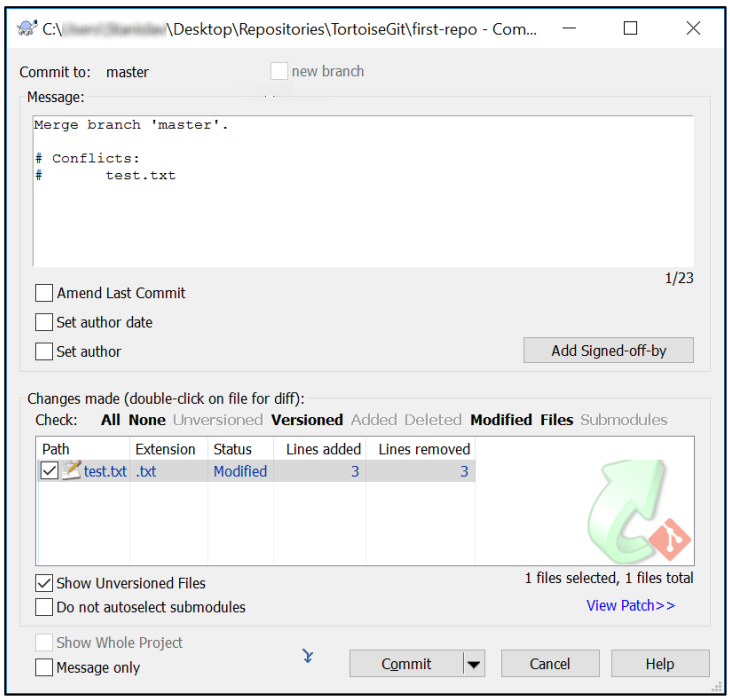


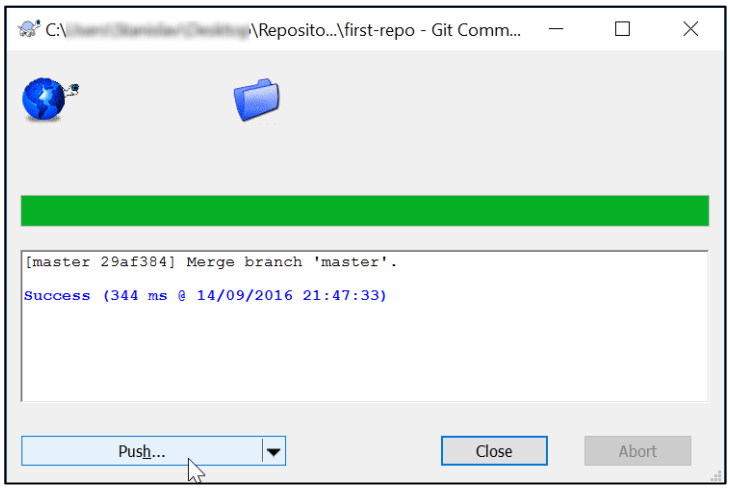
- So you have to **pull** new changes:

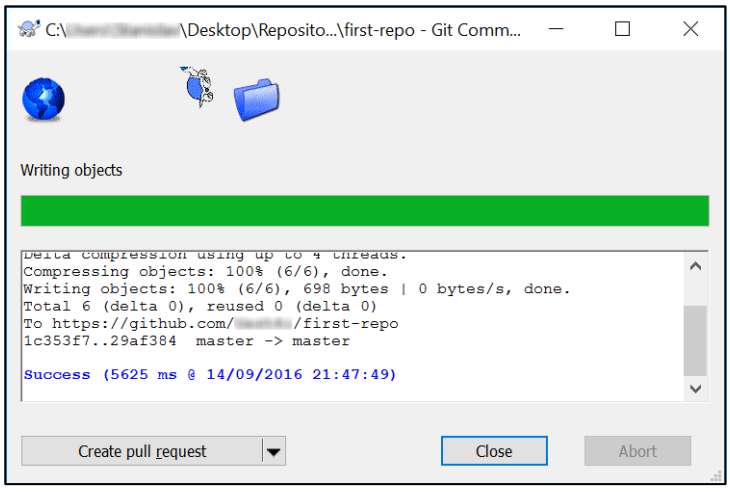


- Note that message: "***Automatic merge failed; fix conflicts…***". We have another conflict, and we have to resolve it like we did earlier, but slightly different:
    - Go on the "**test.txt**" file. You should **open** the **file** and **remove** the same **symbols** that we have previously removed. Then right-click on the file - choose **TortoiseGit** -> **Resolve…** and click it. A dialog window should open. Then you click "**Ok**" to try to **resolve** the conflict.









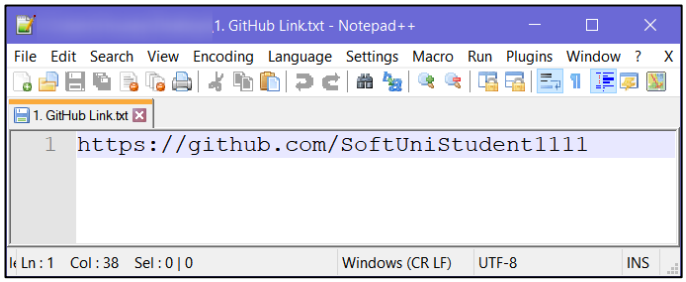
- Now our file is **clean,** and we are ready for our final **commit**!

6. Meet Your Colleagues

It's time to meet a couple of **colleagues** from **SoftUni**. For this exercise, you must submit a **zip** file with all the solutions to the **problems below**.

1. GitHub Profile Link

Create a new **text document** called "**1. GitHub Link.txt**", and put a **link** to your **GitHub profile** inside it. The file should look something like this:



2. GitHub Repository Screenshot

Take a **screenshot** of your **GitHub repository** using something like a snipping tool, then save the file as "**2. GitHub Repo.jpg**".

3. Meet Some Colleagues

First and foremost, look around your colleagues and try to **make acquaintances** with your fellow students. After you meet someone, **note down** the following information about them in a **text document**:

- What is their **name**?
- Where are they **from**?
- What **hobbies/pastimes** do they enjoy?
- Why did they pick **SoftUni**?

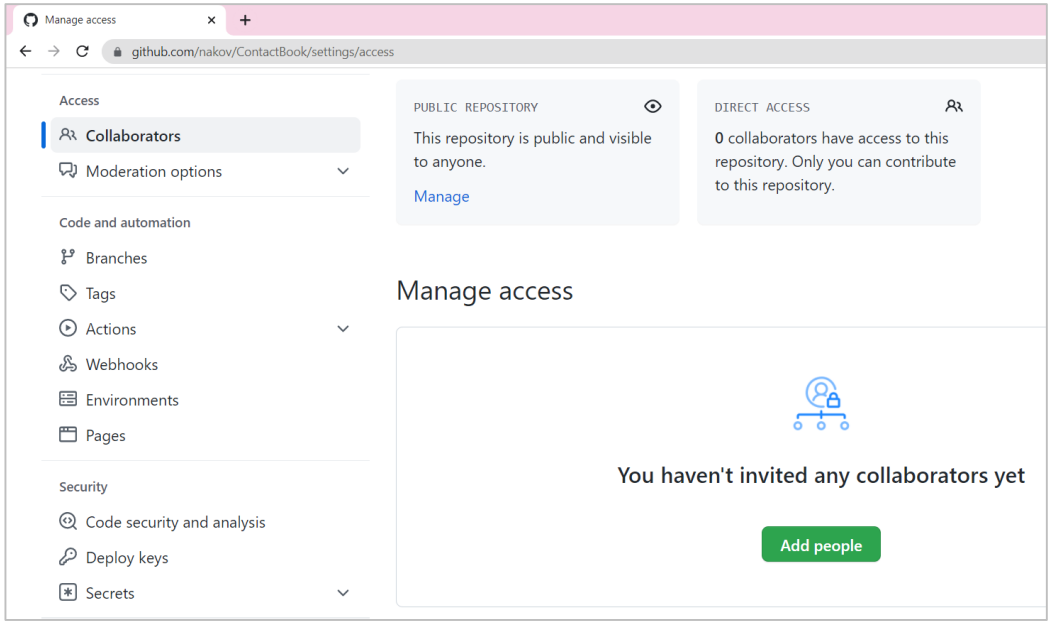
Try to do this with **at least 3** students and also exchange **contact information** with them.

Hopefully, you made a couple of new friends from this exercise.

7. Teamwork

Work into **teams** of (about) 5 students in class:

- Online students work alone or form their teams.
- Each team selects a "**team leader**".
- The team leader **creates a repository** in GitHub, e.g., "**test-repo**".
- The team leader invites his team to the repo:



8. Add a File to GitHub

Team members add a few files:

1. Clone the "**test-repo**" into your computer (if not cloned yet).
2. Create a new file into your working directory:
    - Name the new file "**<your_name>.txt**".
    - Put some text in it the file, e.g., "*My name is …*".
3. Commit the **new file** to your **local repository**.
4. Sync the changes to **upload your file to the remote repo**.
5. Browse the repo from https://github.com/user/repo to check whether your file has been successfully uploaded to GitHub.

9. Create a Git Conflict & Merge

- All team members create a common file, **"config.txt".**
- Each team member adds some settings in **"config.txt"**, e.g.:
    - **name = Peter**
    - **size = 100**
    - **email = peter@dir.bg**
- Each team member **commits** his local changes.
- Each team member **syncs** his changes.
    - The first member will succeed without **conflicts**.
    - The others will have a **conflict** to be merged.
    - **Resolve** the conflict:
        - **Edit** the merged changes + **commit** and **sync** again.