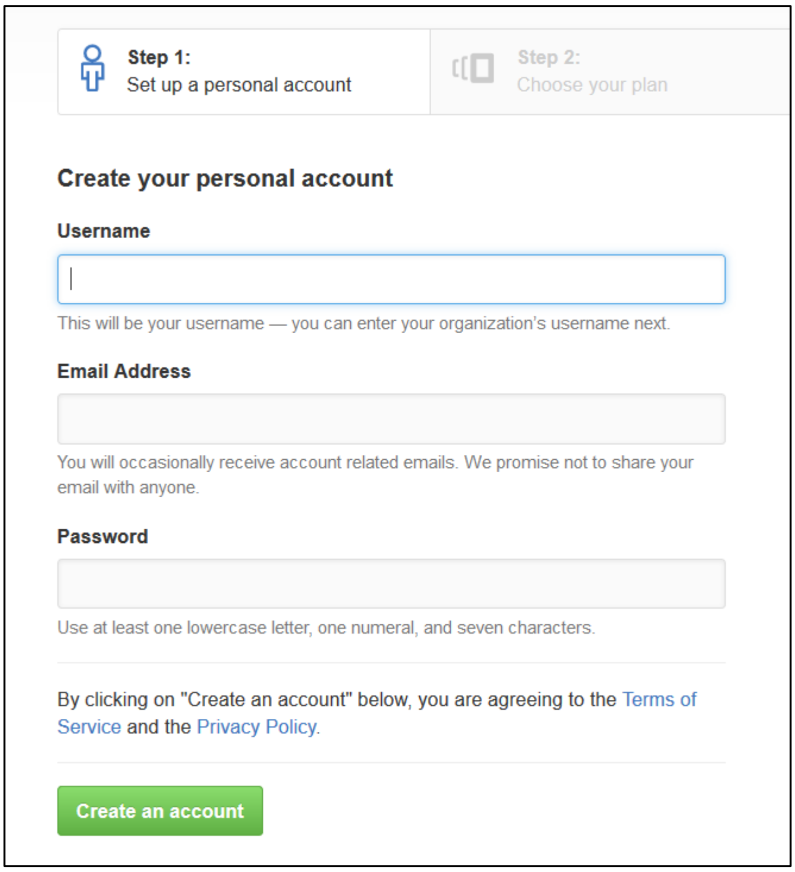
Lab: Git and GitHub

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/


Step 1: Set up a personal account

Step 2: Choose your plan

Create your personal account

Username

This will be your username — you can enter your organization's username next.

Email Address

You will occasionally receive account related emails. We promise not to share your email with anyone.

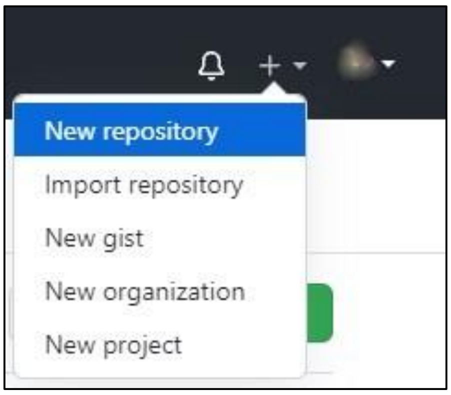
Password

Use at least one lowercase letter, one numeral, and seven characters.

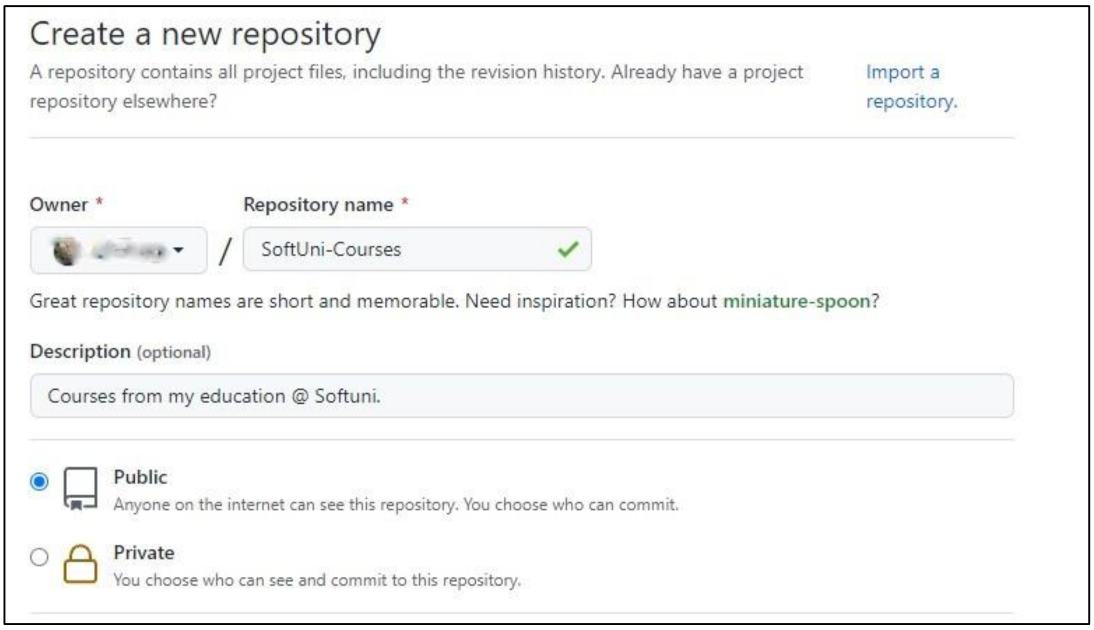
By clicking on "Create an account" below, you are agreeing to the Terms of Service and the Privacy Policy.

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.


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?

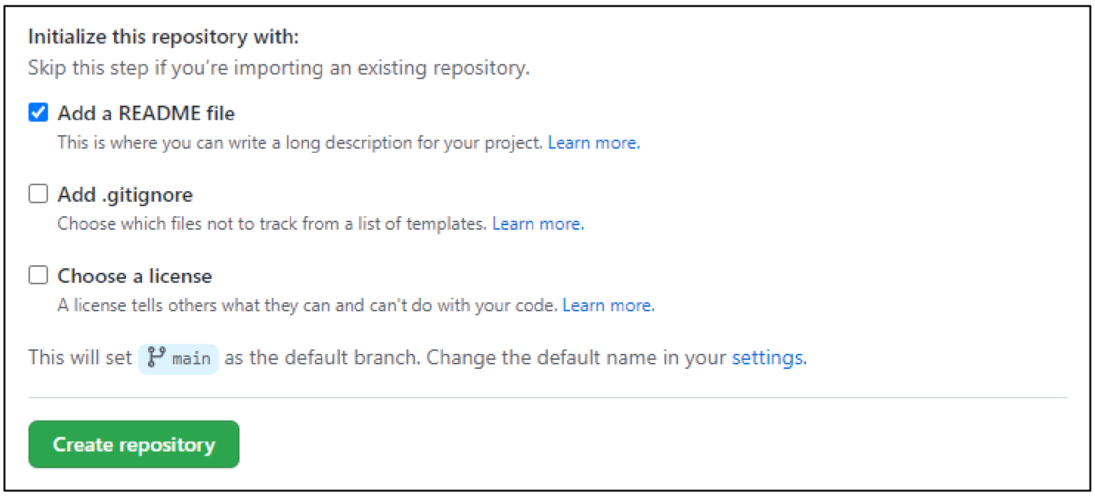
Description (optional)

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:

Skip this step if you're importing an existing repository.

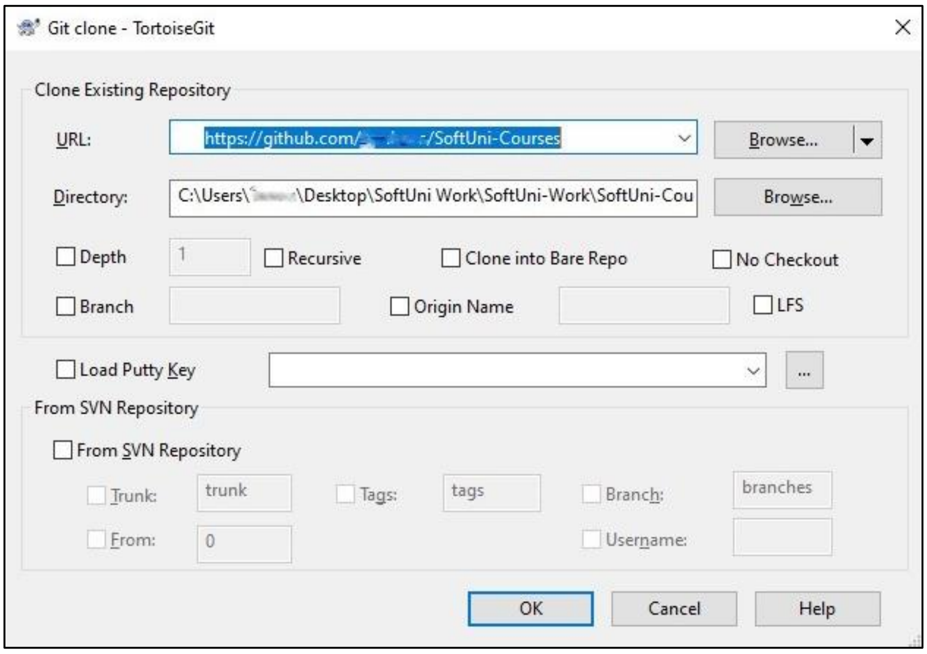
[✓] Add a README file
This is where you can write a long description for your project. Learn more.

[ ] Add .gitignore
Choose which files not to track from a list of templates. Learn more.

[ ] Choose a license
A license tells others what they can and can't do with your code. Learn more.

This will set `main` as the default branch. Change the default name in your settings.

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


Git clone - TortoiseGit

Clone Existing Repository

URL: https://github.com/[illegible]/SoftUni-Courses

Directory: C:\Users\[illegible]\Desktop\SoftUni Work\SoftUni-Work\SoftUni-Cou

[ ] Depth 1 [ ] Recursive [ ] Clone into Bare Repo [ ] No Checkout

[ ] Branch [ ] Origin Name [ ] LFS

[ ] Load Putty Key

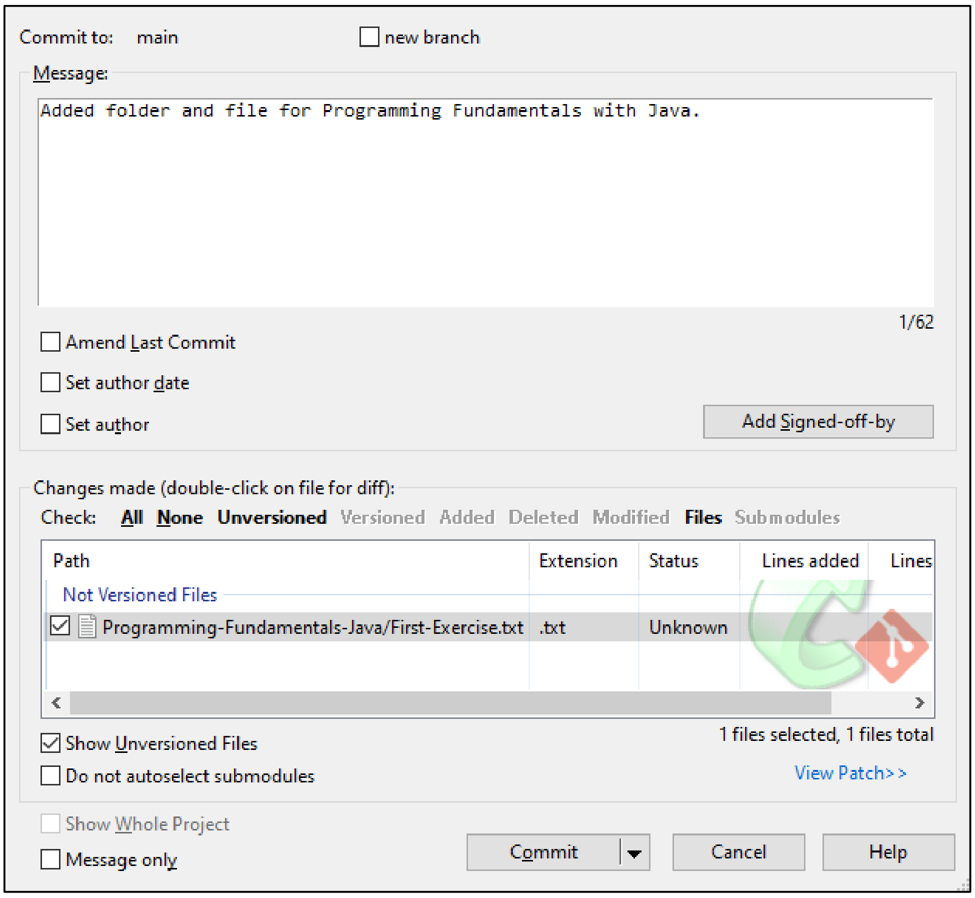
From SVN Repository

[ ] From SVN Repository

[ ] Trunk: trunk [ ] Tags: tags [ ] Branch: branches

[ ] From: 0 [ ] Username:

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".


Commit to: main

Message:

```
Added folder and file for Programming Fundamentals with Java.
```

1/62

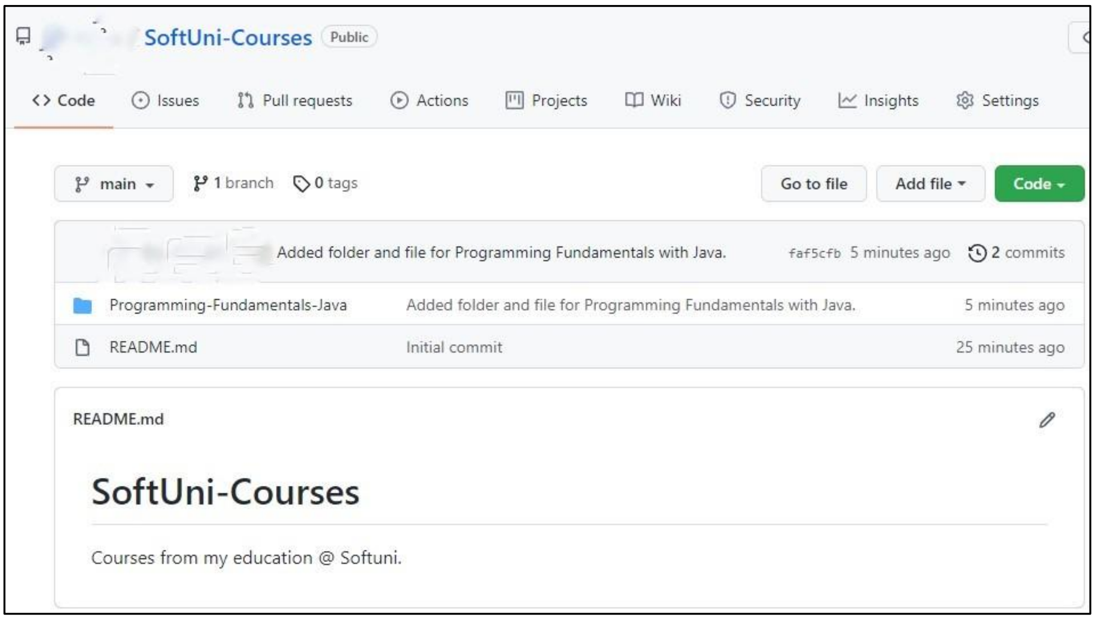
Changes made (double-click on file for diff):

Path	Extension	Status	Lines added	Lines
Not Versioned Files				
☑ Programming-Fundamentals-Java/First-Exercise.txt	.txt	Unknown		

1 files selected, 1 files total

☑ Show Unversioned Files

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



Added folder and file for Programming Fundamentals with Java. faf5cfb 5 minutes ago 2 commits

Programming-Fundamentals-Java	Added folder and file for Programming Fundamentals with Java.	5 minutes ago
README.md	Initial commit	25 minutes ago

README.md

SoftUni-Courses

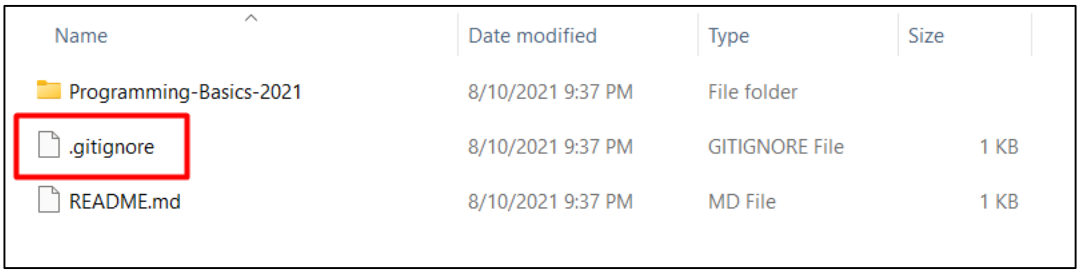
Courses from my education @ Softuni.

We are ready!

III. Configure the .gitignore file

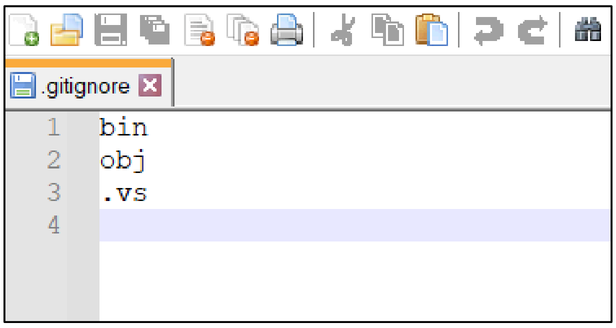
A **.gitignore** file specifies **intentionally untracked files** that Git should ignore. These are all the files that are generated during the project build and compilation. Your repo should keep the source code + documentation + project resources but should ignore all files built from the source code.

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



Name	Date modified	Type	Size
Programming-Basics-2021	8/10/2021 9:37 PM	File folder	
.gitignore	8/10/2021 9:37 PM	GITIGNORE File	1 KB
README.md	8/10/2021 9:37 PM	MD File	1 KB

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



```
bin
obj
.vs
```

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

IV. Conflict + Resolve

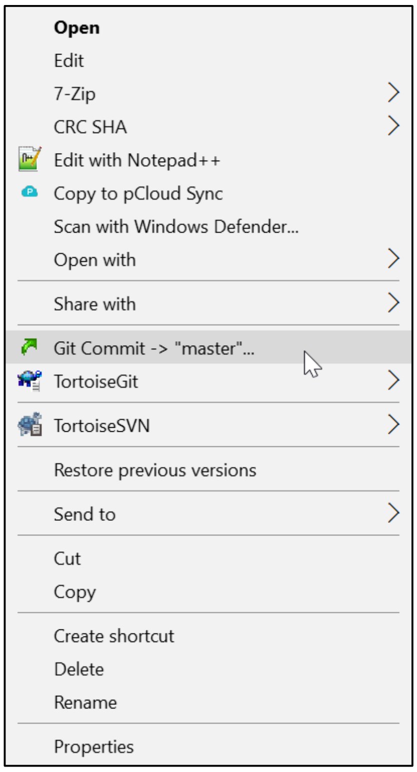
1. Make a Conflict

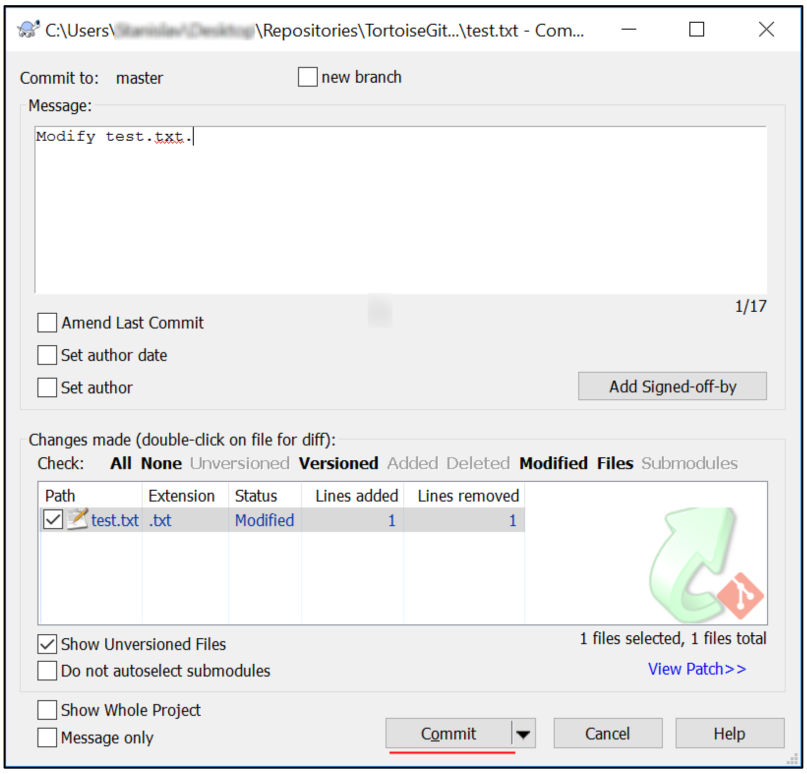
Update the content in both directories separately:

- 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:




C:\Users\[redacted]\Repositories\TortoiseGit...\test.txt - Com...

Commit to: master ☐ new branch

Message:

```
Modify test.txt.
```

1/17

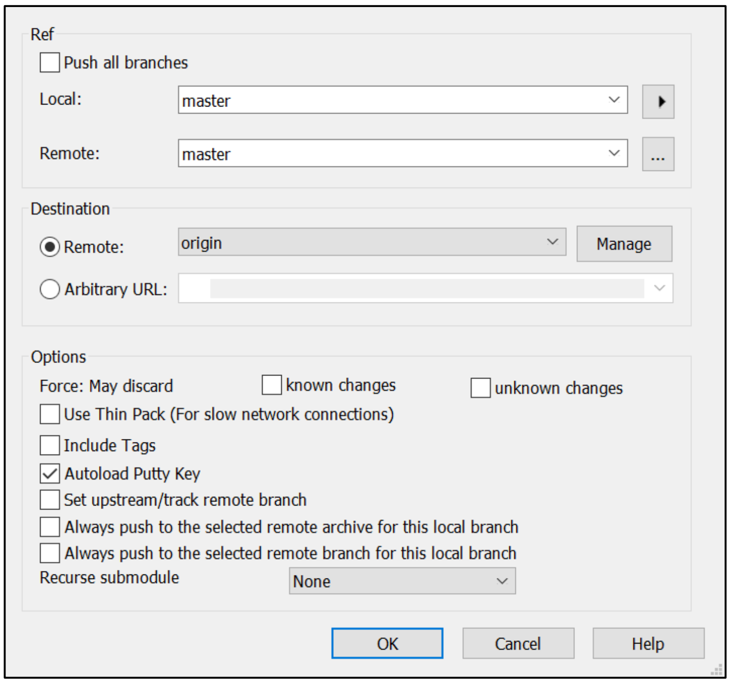
☐ Amend Last Commit
☐ Set author date
☐ Set author

Changes made (double-click on file for diff):

Check: **All** **None** Unversioned **Versioned** Added Deleted **Modified** **Files** Submodules

Path	Extension	Status	Lines added	Lines removed
✓ test.txt	.txt	Modified	1	1

☑ Show Unversioned Files
☐ Do not autoselect submodules
1 files selected, 1 files total
☐ Show Whole Project
☐ Message only


Ref

☐ Push all branches

Local: master

Remote: master

Destination

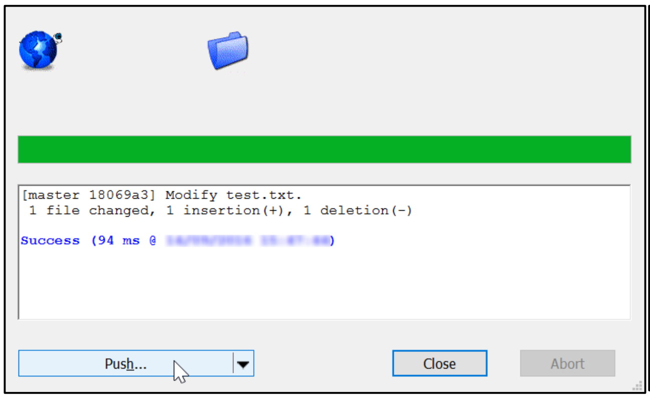
● Remote: origin

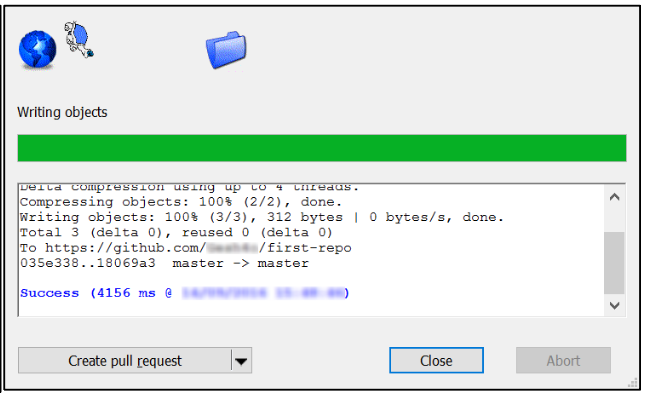
○ Arbitrary URL:

Options

Force: May discard ☐ known changes ☐ unknown changes
☐ Use Thin Pack (For slow network connections)
☐ Include Tags
☑ Autoload Putty Key
☐ Set upstream/track remote branch
☐ Always push to the selected remote archive for this local branch
☐ Always push to the selected remote branch for this local branch

Recurse submodule: None

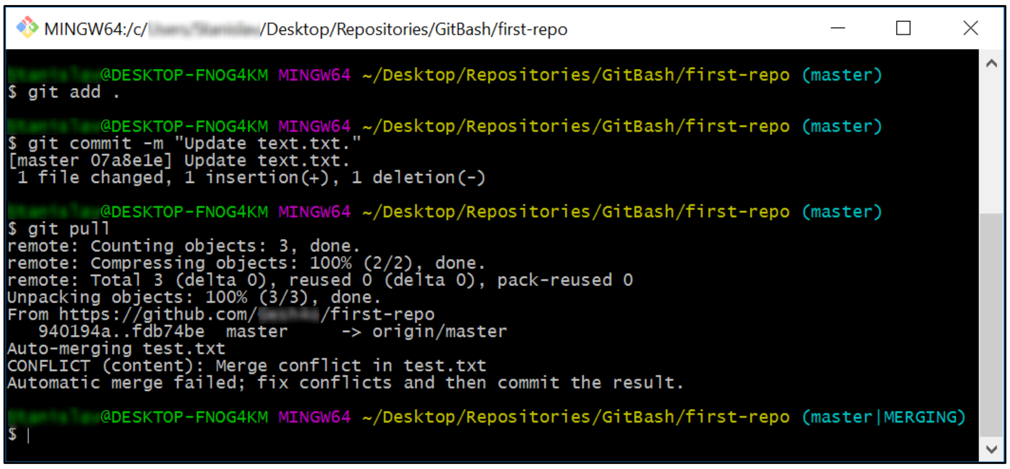

```
[master 18069a3] Modify test.txt.
 1 file changed, 1 insertion(+), 1 deletion(-)

Success (94 ms @ [redacted])
```


Writing objects

```
Delta compression using up to 4 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 312 bytes | 0 bytes/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/[redacted]/first-repo
035e338..18069a3  master -> master

Success (4156 ms @ [redacted])
```

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**"



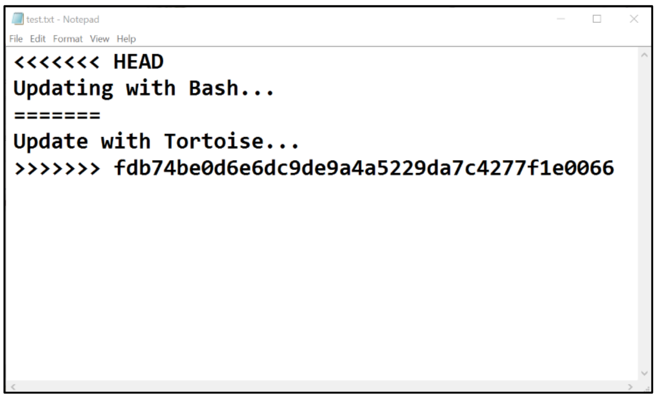
MINGW64:/c/[redacted]/Desktop/Repositories/GitBash/first-repo

```
[redacted]@DESKTOP-FNOG4KM MINGW64 ~/Desktop/Repositories/GitBash/first-repo (master)
$ git add .

[redacted]@DESKTOP-FNOG4KM MINGW64 ~/Desktop/Repositories/GitBash/first-repo (master)
$ git commit -m "Update text.txt."
[master 07a8e1e] Update text.txt.
 1 file changed, 1 insertion(+), 1 deletion(-)

[redacted]@DESKTOP-FNOG4KM MINGW64 ~/Desktop/Repositories/GitBash/first-repo (master)
$ git pull
remote: Counting objects: 3, done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.
From https://github.com/[redacted]/first-repo
   940194a..fdb74be  master     -> origin/master
Auto-merging test.txt
CONFLICT (content): Merge conflict in test.txt
Automatic merge failed; fix conflicts and then commit the result.

[redacted]@DESKTOP-FNOG4KM MINGW64 ~/Desktop/Repositories/GitBash/first-repo (master|MERGING)
$ |
```

4. Merge a Conflict

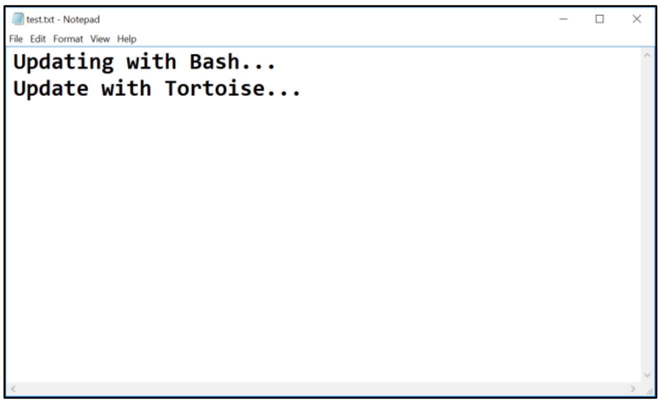
Now you have a "**merge conflict**" which you have to resolve. The **"git pull"** command **automatically** created it because the remote repository at GitHub had a newer version for some files of your code.

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



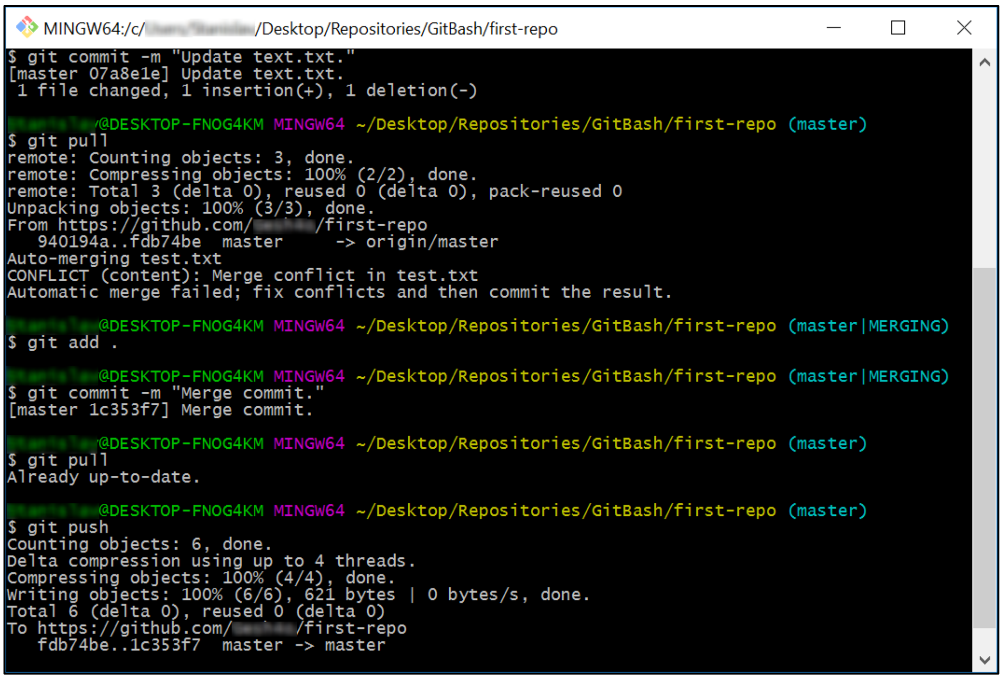
```
<<<<<<< HEAD
Updating with Bash...
=======
Update with Tortoise...
>>>>>>> fdb74be0d6e6dc9de9a4a5229da7c4277f1e0066
```

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



```
Updating with Bash...
Update with Tortoise...
```

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

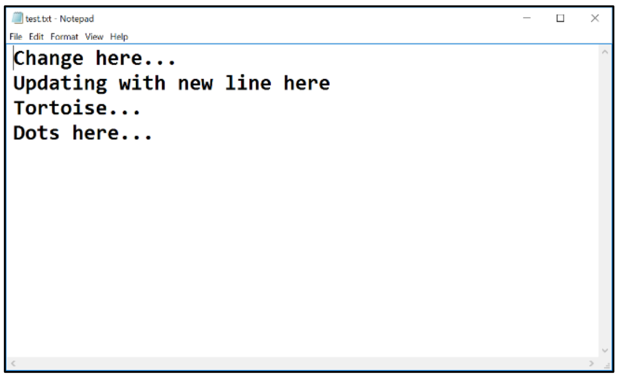

MINGW64:/c/[redacted]/Desktop/Repositories/GitBash/first-repo

```
$ git commit -m "Update text.txt."
[master 07a8e1e] Update text.txt.
 1 file changed, 1 insertion(+), 1 deletion(-)

[redacted]@DESKTOP-FNOG4KM MINGW64 ~/Desktop/Repositories/GitBash/first-repo (master)
$ git pull
remote: Counting objects: 3, done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.
From https://github.com/[redacted]/first-repo
   940194a..fdb74be  master     -> origin/master
Auto-merging test.txt
CONFLICT (content): Merge conflict in test.txt
Automatic merge failed; fix conflicts and then commit the result.

[redacted]@DESKTOP-FNOG4KM MINGW64 ~/Desktop/Repositories/GitBash/first-repo (master|MERGING)
$ git add .

[redacted]@DESKTOP-FNOG4KM MINGW64 ~/Desktop/Repositories/GitBash/first-repo (master|MERGING)
$ git commit -m "Merge commit."
[master 1c353f7] Merge commit.

[redacted]@DESKTOP-FNOG4KM MINGW64 ~/Desktop/Repositories/GitBash/first-repo (master)
$ git pull
Already up-to-date.

[redacted]@DESKTOP-FNOG4KM MINGW64 ~/Desktop/Repositories/GitBash/first-repo (master)
$ git push
Counting objects: 6, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (4/4), done.
Writing objects: 100% (6/6), 621 bytes | 0 bytes/s, done.
Total 6 (delta 0), reused 0 (delta 0)
To https://github.com/[redacted]/first-repo
   fdb74be..1c353f7  master -> master
```

5. Merge Changes and Push to GitHub

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

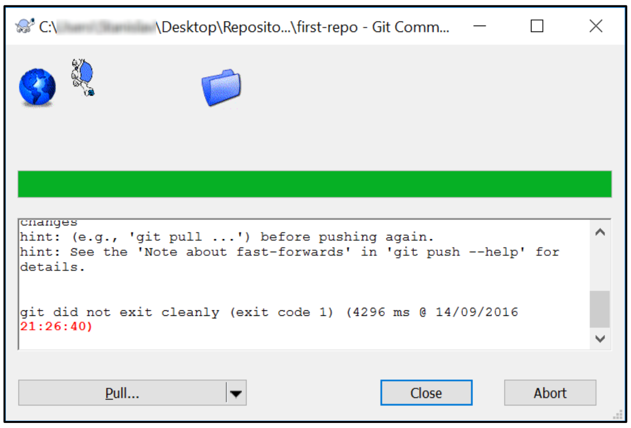
- Make additional changes to the file **test.txt** and **commit** them.



```
Change here...
Updating with new line here
Tortoise...
Dots here...
```

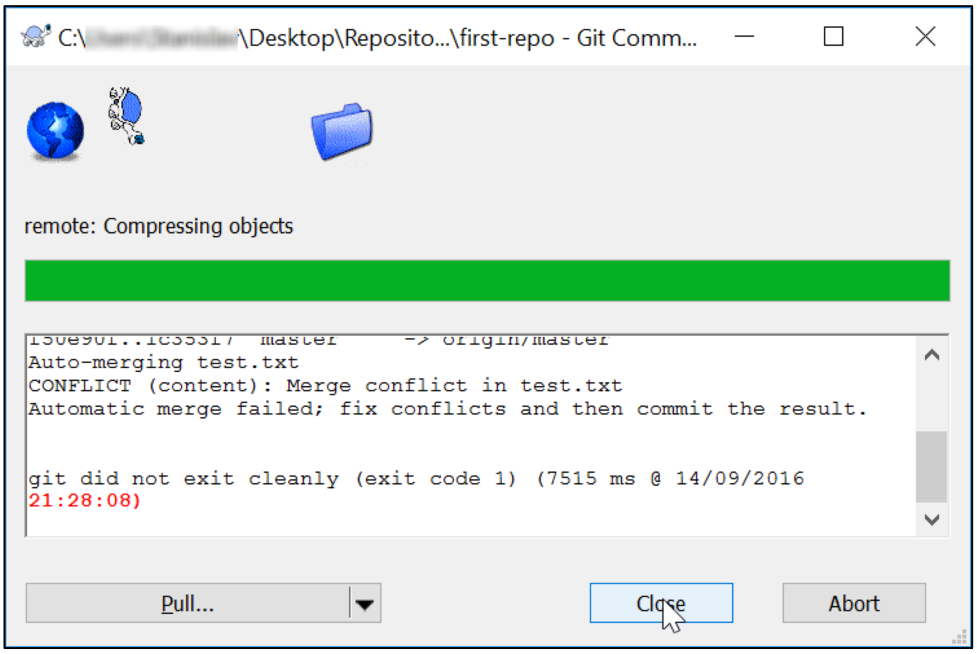
***Note** that if your changes are simple (e.g. just a new file is added), TortoiseGit may **automatically** merge them.

- 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.



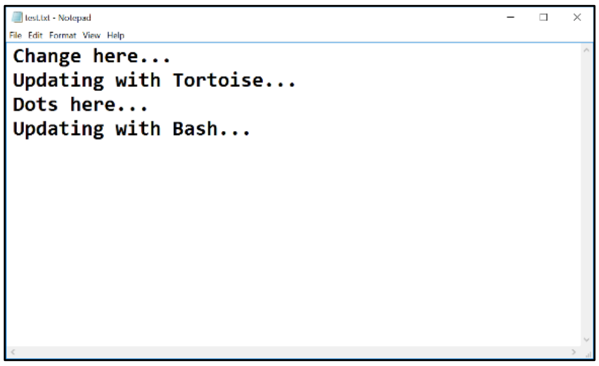
```
changes
hint: (e.g., 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for
details.

git did not exit cleanly (exit code 1) (4296 ms @ 14/09/2016
21:26:40)
```

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

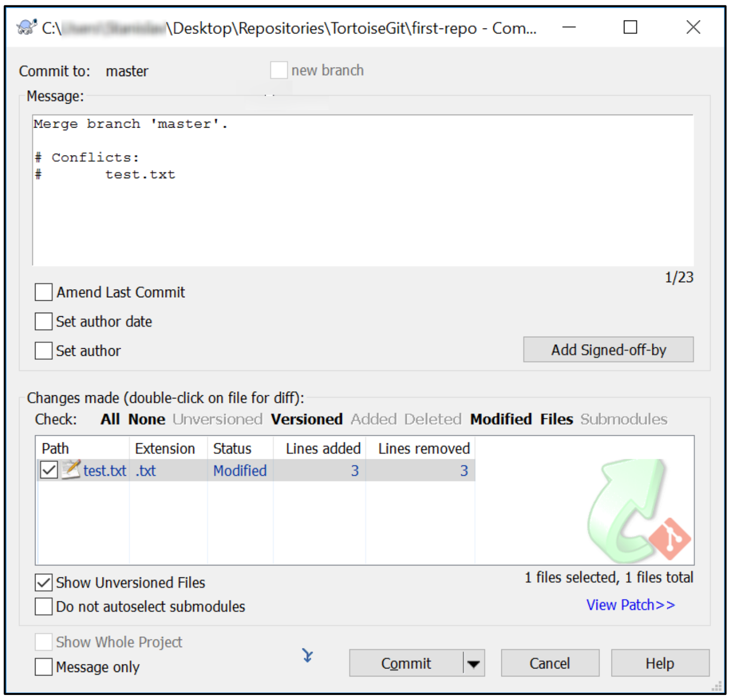

C:\[redacted]\Desktop\Reposito...\first-repo - Git Comm...
remote: Compressing objects
Pull...
Close
Abort

```
150e901..1c53317  master     -> origin/master
Auto-merging test.txt
CONFLICT (content): Merge conflict in test.txt
Automatic merge failed; fix conflicts and then commit the result.


git did not exit cleanly (exit code 1) (7515 ms @ 14/09/2016
21:28:08)
```

- 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.

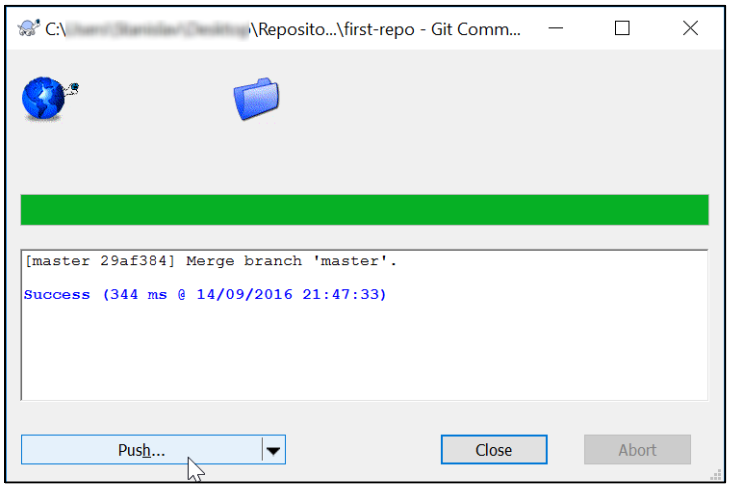


```
Change here...
Updating with Tortoise...
Dots here...
Updating with Bash...
```

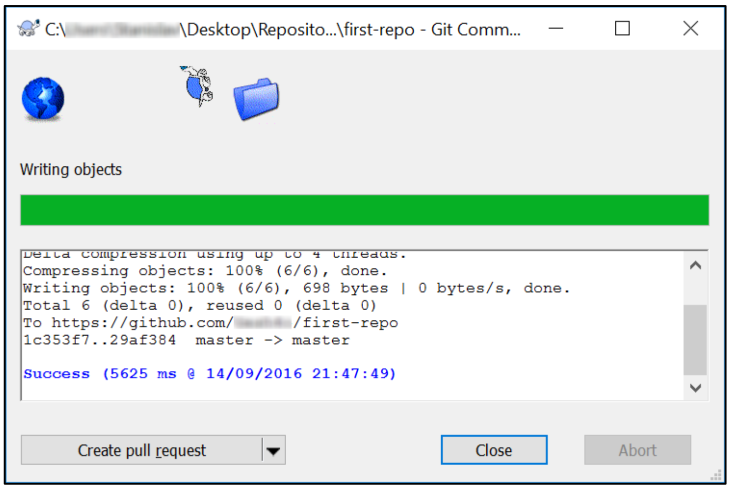

C:\[redacted]\Desktop\Repositories\TortoiseGit\first-repo - Com...
Commit to: master
new branch
Message:
1/23
Amend Last Commit
Set author date
Set author
Add Signed-off-by
Changes made (double-click on file for diff):
Check: All None Unversioned Versioned Added Deleted Modified Files Submodules
Show Unversioned Files
Do not autoselect submodules
1 files selected, 1 files total
View Patch>>
Show Whole Project
Message only
Commit
Cancel
Help

```
Merge branch 'master'.

# Conflicts:
#       test.txt
```

Path	Extension	Status	Lines added	Lines removed
✓ test.txt	.txt	Modified	3	3



```
[master 29af384] Merge branch 'master'.

Success (344 ms @ 14/09/2016 21:47:33)
```



```
Delta compression using up to 4 threads.
Compressing objects: 100% (6/6), done.
Writing objects: 100% (6/6), 698 bytes | 0 bytes/s, done.
Total 6 (delta 0), reused 0 (delta 0)
To https://github.com/[redacted]/first-repo
1c353f7..29af384  master -> master

Success (5625 ms @ 14/09/2016 21:47:49)
```

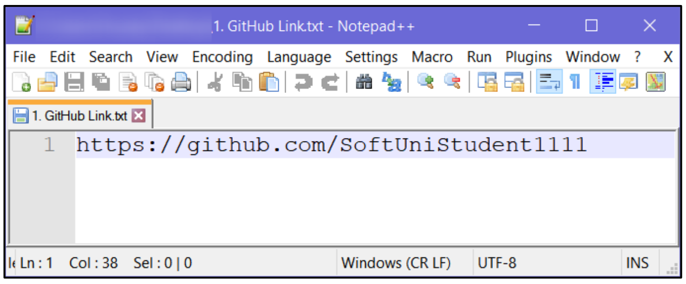
- Now our file is **clean,** and we are ready for our final **commit**!

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:


1. GitHub Link.txt - Notepad++
File Edit Search View Encoding Language Settings Macro Run Plugins Window ? X
1. GitHub Link.txt
Ln : 1 Col : 38 Sel : 0 | 0 Windows (CR LF) UTF-8 INS

```
https://github.com/SoftUniStudent1111
```

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**?

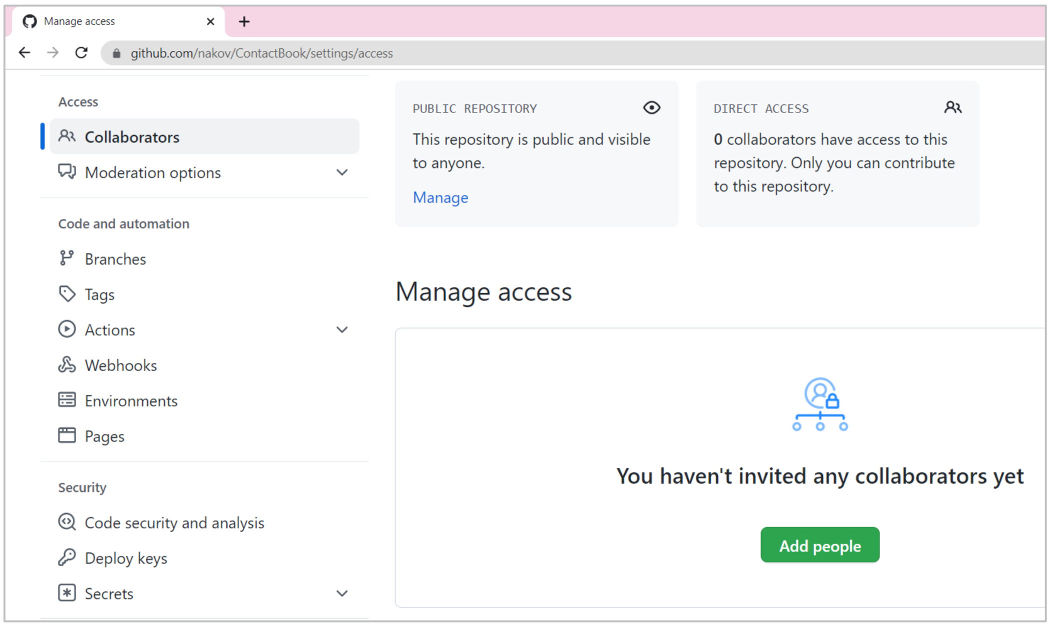
Try to do this with **at least 3** students and also exchange **contact information** with them.

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:



PUBLIC REPOSITORY

This repository is public and visible to anyone.

Manage

DIRECT ACCESS

0 collaborators have access to this repository. Only you can contribute to this repository.

Manage access

You haven't invited any collaborators yet

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.