

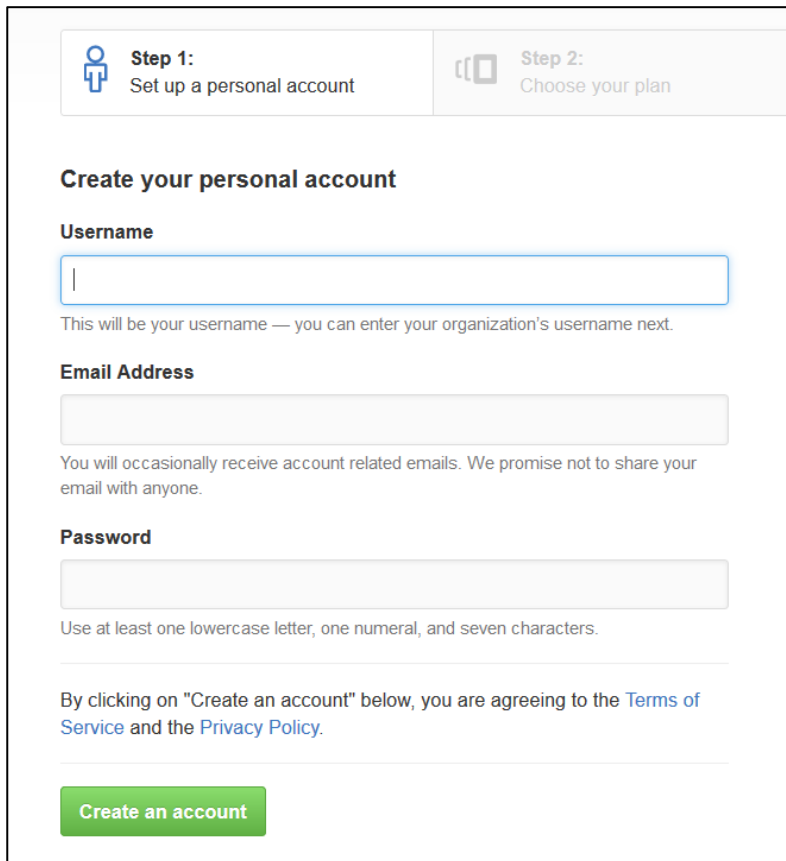
Lab: Git and GitHub

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

I. Create a GitHub Developer Profile

1. Create a GitHub Profile

Register for a free **developer account** at GitHub: <http://github.com>. Submit your developer profile's URL as output of this homework.



The screenshot shows the GitHub account creation interface. At the top, there are two steps: 'Step 1: Set up a personal account' (active) and 'Step 2: Choose your plan'. The main heading is 'Create your personal account'. Below this, there are three input fields: 'Username', 'Email Address', and 'Password'. The 'Username' field has a placeholder text: 'This will be your username — you can enter your organization's username next.' The 'Email Address' field has a placeholder text: 'You will occasionally receive account related emails. We promise not to share your email with anyone.' The 'Password' field has a placeholder text: 'Use at least one lowercase letter, one numeral, and seven characters.' Below the input 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'.


II. Creating a Repo + Conflict + Resolve

2. Create a GitHub Repository

- New repository form: <https://github.com/new>.
- Choose a name for the repo, e.g. "**first-repo**". Make sure to "Initialize this repository with a README".

Create a new repository

A repository contains all project files, including the revision history.

Owner:  / Repository name: ✓

Great repository names are short and memorable. Need inspiration? How about **fantastic-octo-computing-machine?**

Description (optional):

☒ **Public**
Anyone can see this repository. You choose who can commit.

☐ **Private**
You choose who can see and commit to this repository.

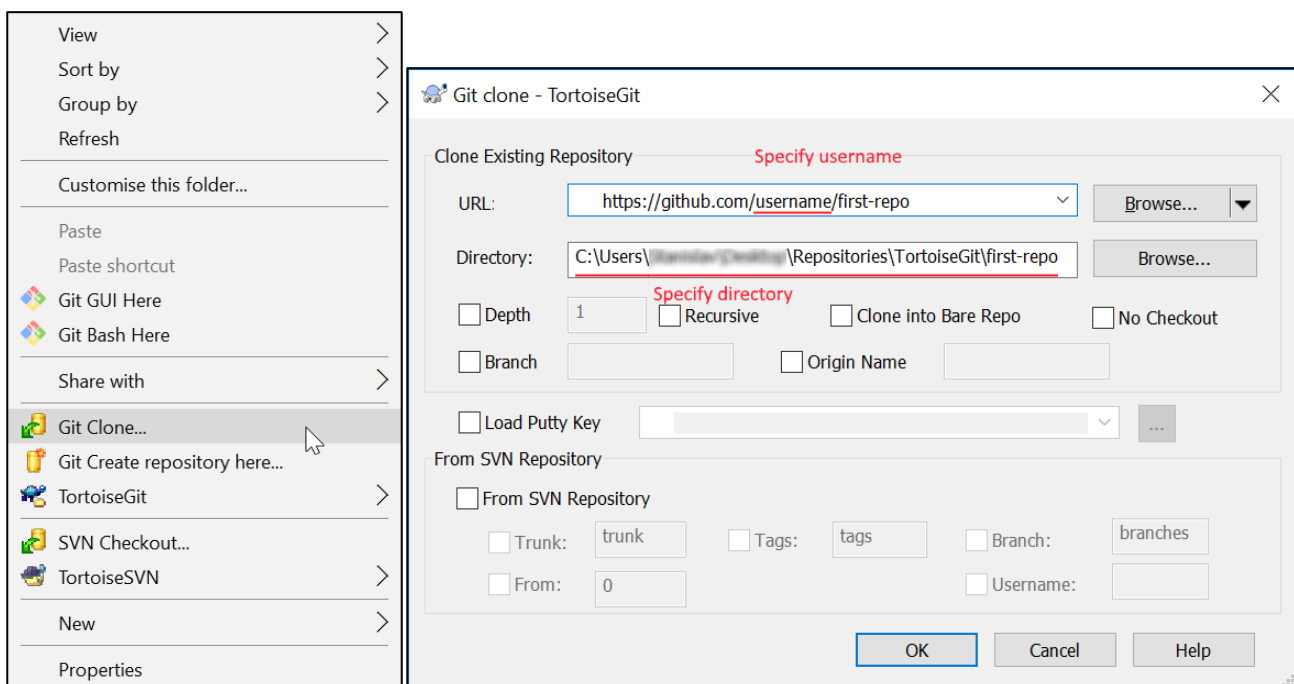
☒ **Initialize this repository with a README**
This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: None ▾ Add a license: None ▾ [i](#)

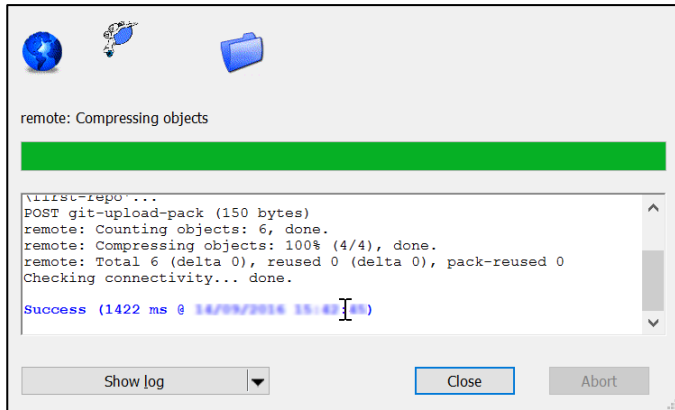
3. Clone a Repository Twice

Clone that repository on two different places on your personal device.

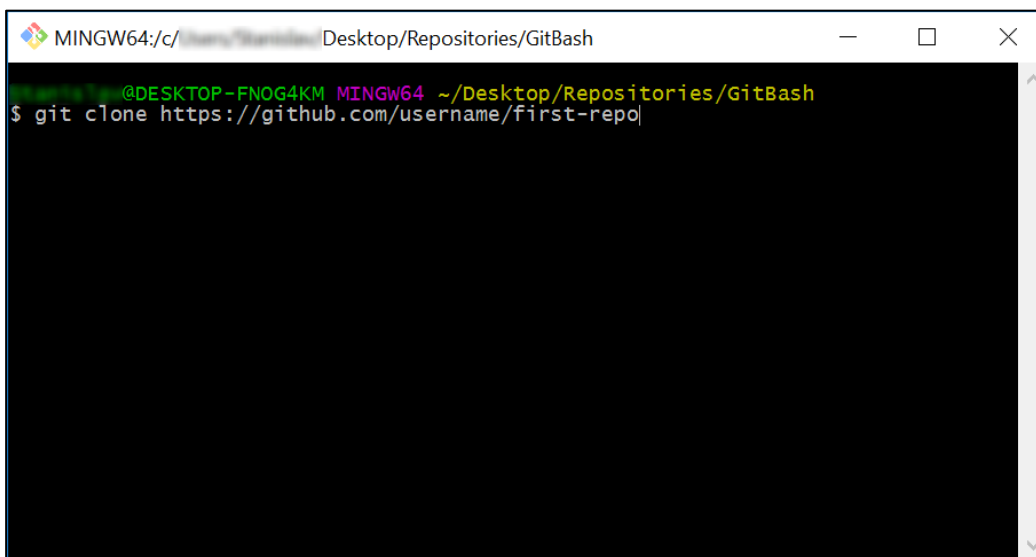
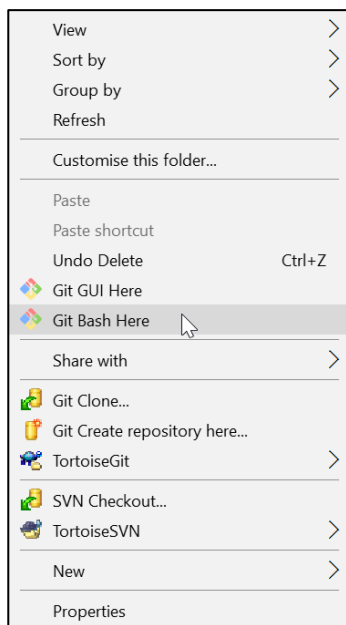
- Use Git **clone** for cloning with **TortoiseGit**.
 - Go in the desired directory, right click on blank space anywhere in the folder and copy the link of your repository.



- The result should be something like this:



- Use "**git clone**" command for cloning with **GitBash**.
 - Go to the desired **directory**, right click on blank space anywhere in the folder, select "**Git Bash here**" and type "**git clone**" command followed by the link of your repository.



- The result should be something like this:

```

MINGW64: /C:/Users/.../Desktop/Repositories/GitBash
$ git clone https://github.com/.../first-repo
Cloning into 'first-repo'...
remote: Counting objects: 9, done.
remote: Compressing objects: 100% (6/6), done.
remote: Total 9 (delta 0), reused 3 (delta 0), pack-reused 0
Unpacking objects: 100% (9/9), done.
Checking connectivity... done.

MINGW64: /C:/Users/.../Desktop/Repositories/GitBash
$

```

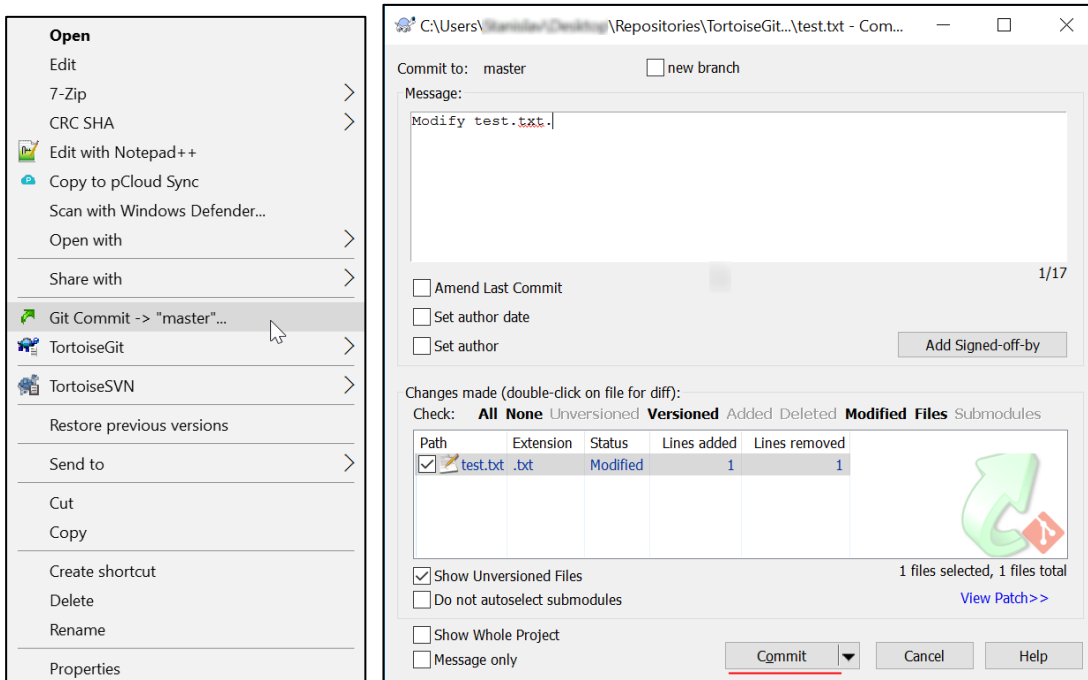
4. Make a Conflict

Update content in both directories differently:

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

5. Upload Changes

Upload Your Changes from **TortoiseGit** Clone. You can use TortoiseGit's "**Git Commit...**":



Ref

☐ Push all branches

Local: ▶

Remote: ...

Destination

☒ Remote: Manage

☐ 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

OK Cancel Help

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/first-repo

035e338..18069a3 master -> master

Success (94 ms @ 18/06/2016 15:47:48)

Push... Close Abort

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/first-repo

035e338..18069a3 master -> master

Success (4156 ms @ 18/06/2016 15:48:48)

Create pull request Close Abort

6. Update Bash Clone

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

```
MINGW64:/c:/Users/.../Desktop/Repositories/GitBash/first-repo
$ git add .
$ git commit -m "Update text.txt."
[master 07a8e1e] Update text.txt.
1 file changed, 1 insertion(+), 1 deletion(-)
$ 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/.../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.
$
```

7. Merge Conflict

Now you have merge conflict which you have to resolve

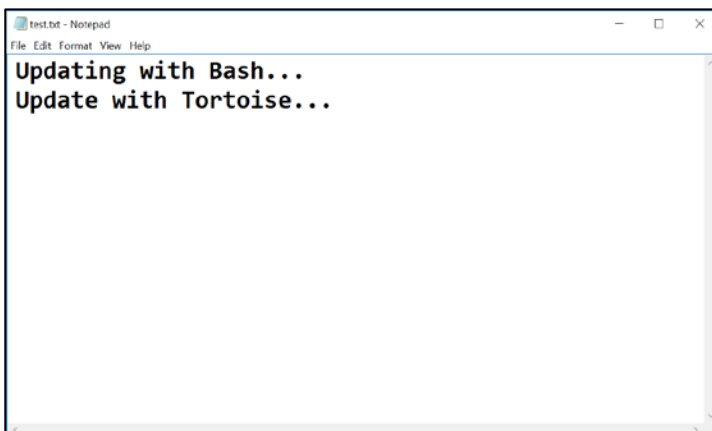
- Open the **test.txt** file in your **GitBash** clone, it should look like this:



```
test.txt - Notepad
File Edit Format View Help

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

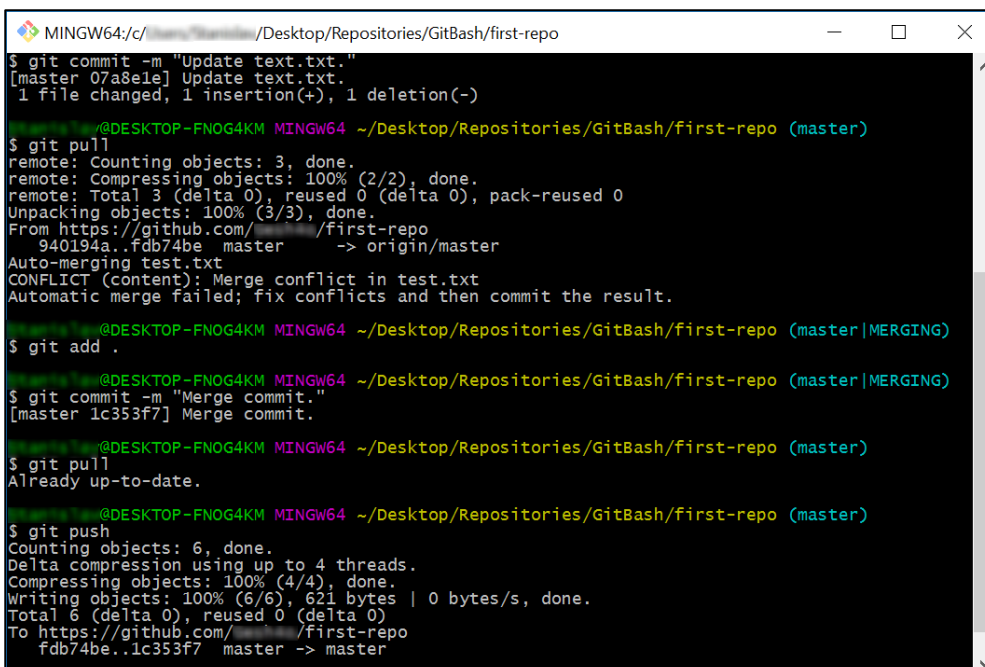
- Remove the HEAD, =====, <<<<<<, >>>>>> symbols and save the file.



```
test.txt - Notepad
File Edit Format View Help

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.



```
MINGW64/c:/Users/.../Desktop/Repositories/GitBash/first-repo
$ git commit -m "Update text.txt."
[master 07a8e1e] Update text.txt.
1 file changed, 1 insertion(+), 1 deletion(-)

$ 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/.../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.

$ git add .
$ git commit -m "Merge commit."
[master 1c353f7] Merge commit.

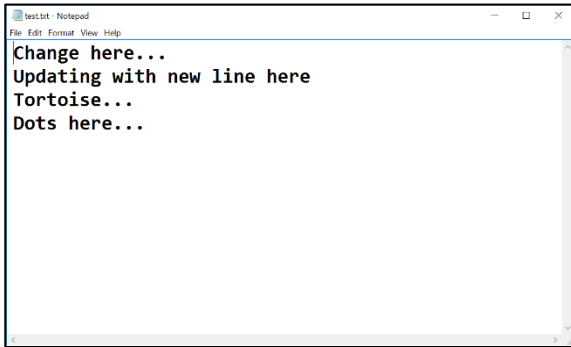
$ git pull
Already up-to-date.

$ 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/.../first-repo
  fdb74be..1c353f7  master -> master
```

8. Merge Changes and Push to GitHub

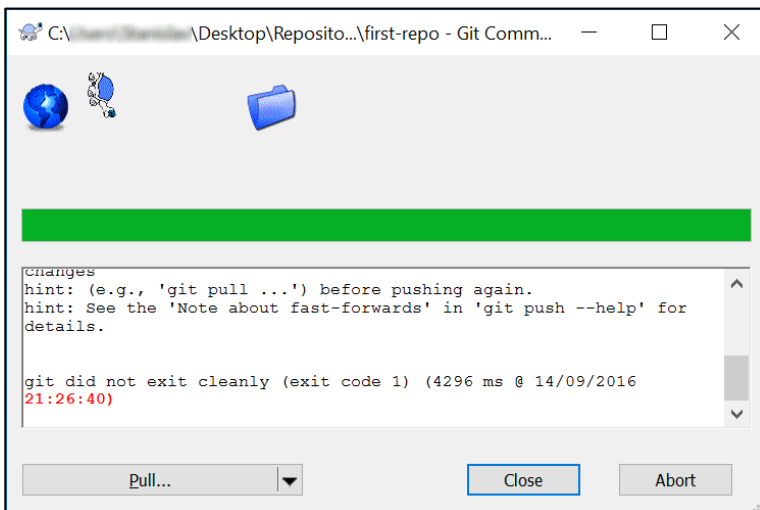
You have updated the content of your remote repository, now try to update your TortoiseGit clone

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

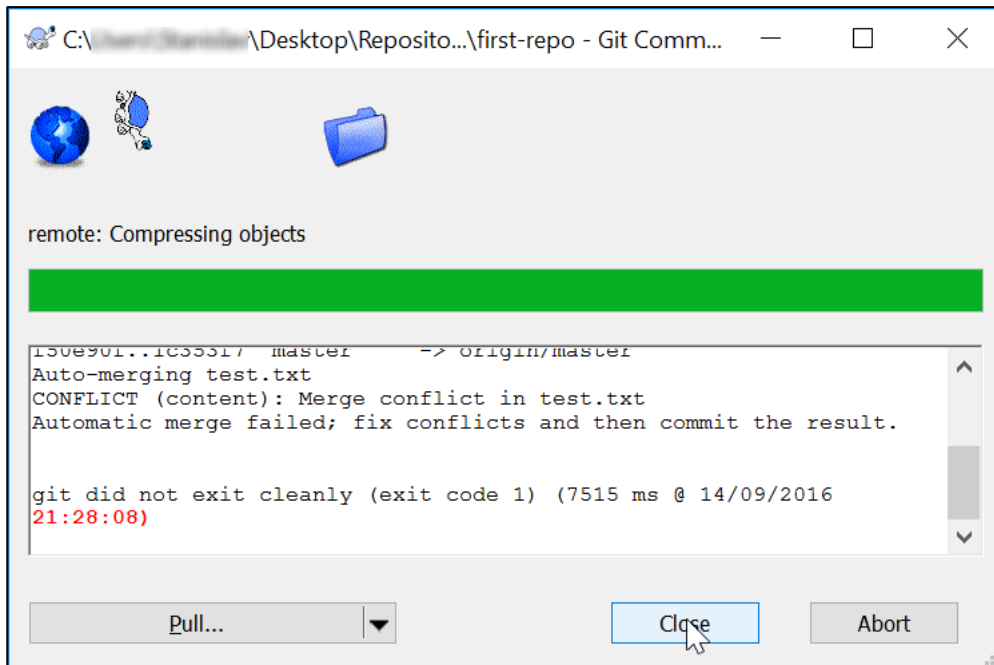


***Note** that if you make changes too simple TortoiseGit may **automatically** merge them.

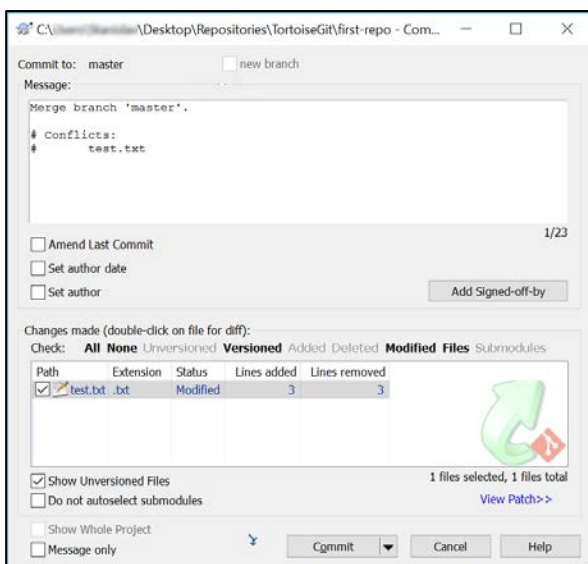
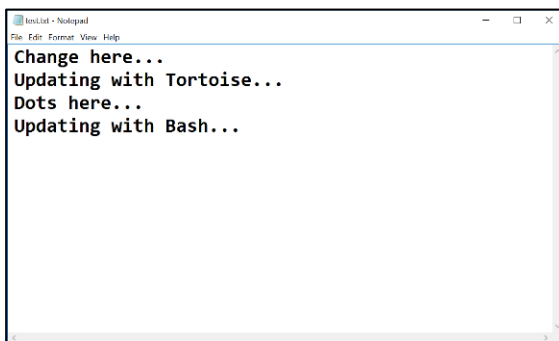
- Now try to **push**. It turns out that we have our **remote** repository **updated** (the merge commit) and we do not have these changes on our **local** repository.

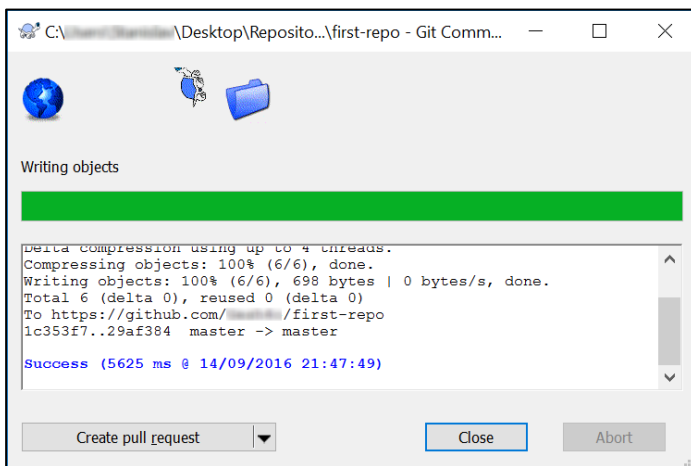
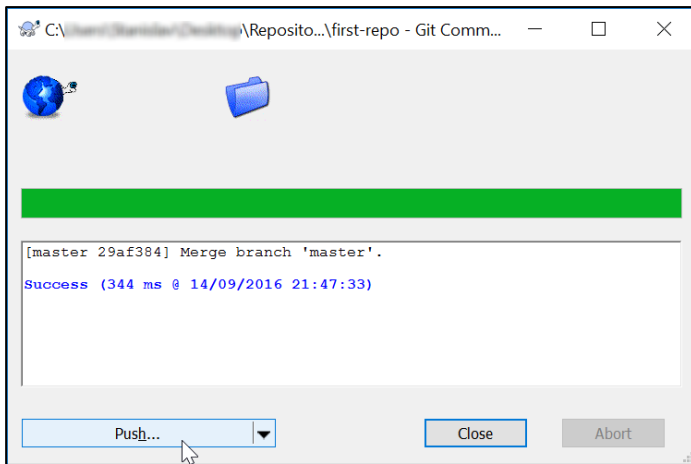


- So we have to **pull** new changes:



- Note that message: "Automatic merge failed; fix conflicts...". We have another conflict and we have to resolve it like we did earlier but small difference:
 - 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**" in order to try to **resolve** the conflict.





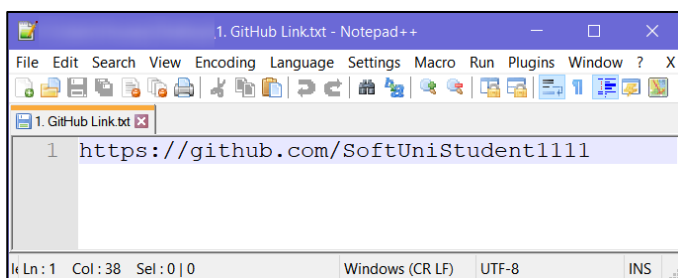
- Now our file is **clean** and we are ready for our final **commit**!

III. Meet Your Colleagues

It's time to meet a couple of **colleagues** at **SoftUni**. For this exercise, you must submit a **zip** file with all the solutions from 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 [snipping tool](#), then save the file as "**2. GitHub Repo.jpg**".

3. Meet Some Colleagues

First and foremost, look around the hall 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 new friends from this exercise. 😊

4. Upload Homework to SoftUni

Put all of the text files and screenshots you created in a **zip file** and **send it as homework** on the **SoftUni** site.

IV. Teamwork

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

- Online students work alone or form own teams.
- Each team selects a "**team leader**".

The team leader **creates a repository** in GitHub e.g. "**test-repo**":

5. 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 is successfully uploaded in GitHub.

6. 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.