



Упражнения: Git и GitHub

Задачи за упражнения и домашни върху [курса "Увод в програмирането" @ СофтУни](#).

I. Създаване на профил на разработчик в GitHub

1. Създайте си профил в GitHub

Регистрирайте се за безплатен профил на разработчик в GitHub: <http://github.com>. Предайте URL-а на вашия профил като решение на това домашно.

 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](#).

Create an account

II. Създайте хранилище (Репо) + Конфликт + Разрешаване

2. Създайте GitHub хранилище

- Създайте ново хранилище от: <https://github.com/new>.
- Изберете име на хранилището, например. **"first-repo"**. Убедете се, че сте избрали опцията **"Initialize this repository with a README"**.
- Качете прост **"test.txt"** файл с някакво съдържание за проба.

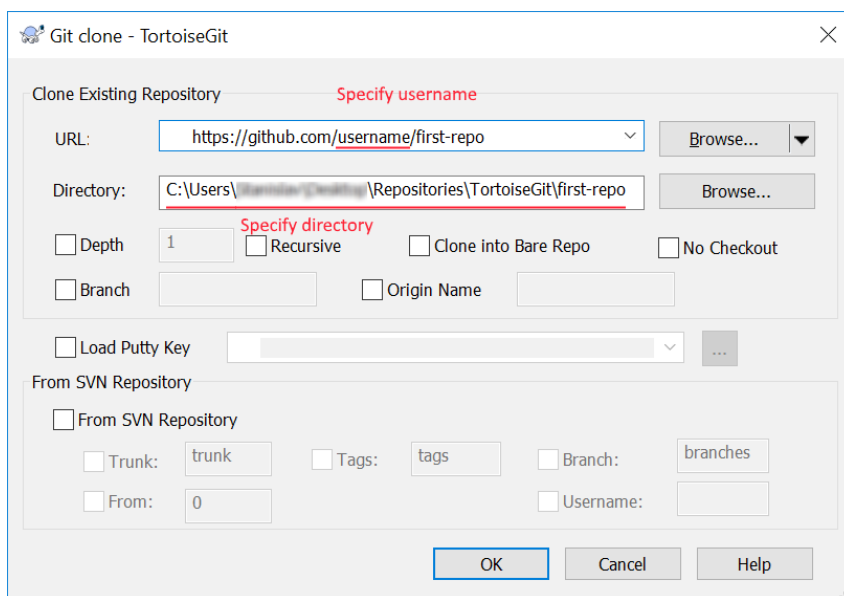
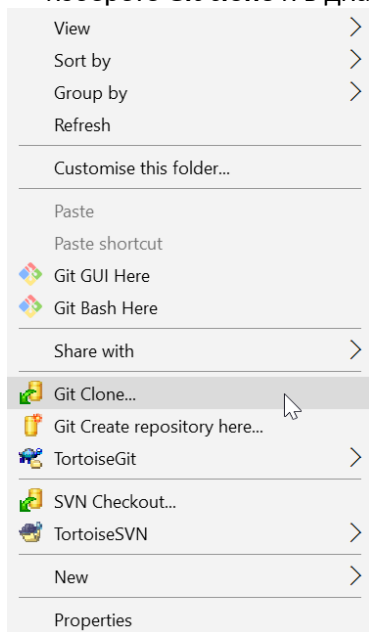
Create new file Upload files

- Превъртете до дъното на страницата, укажете commit съобщение и натиснете [Commit changes]

3. Клонирайте хранилището два пъти

Клонирайте току-що създаденото хранилище на **две различни места** на вашето устройство.

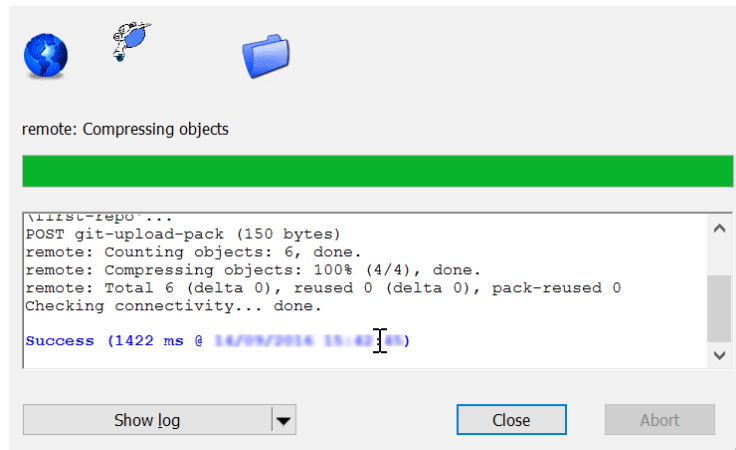
- Използвайте Git **clone** за клониране с **TortoiseGit**.
 - Отидете в избраната от вас папка, щракнете с десен бутон някъде на празно място в папката, изберете **Git clone** и в диалоговия прозорец, който се появи, поставете URL адреса на вашето



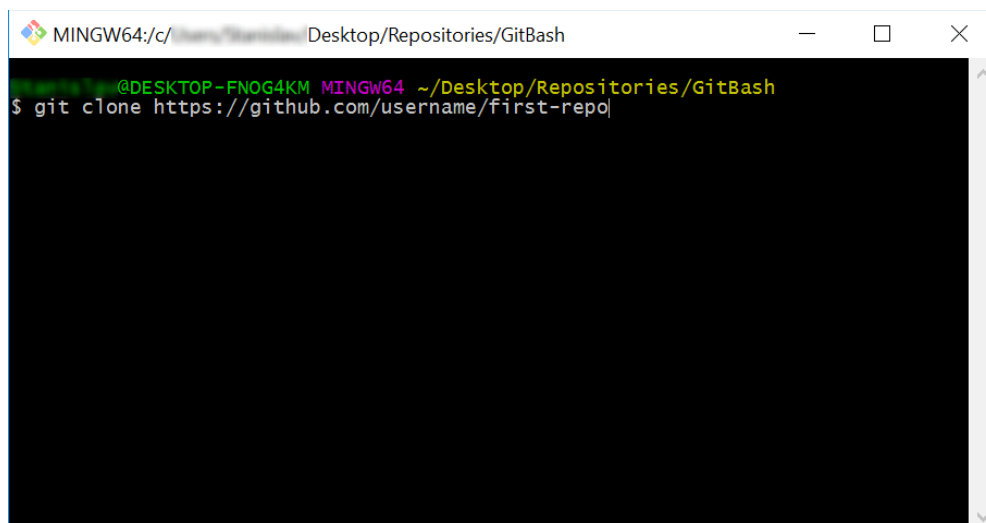
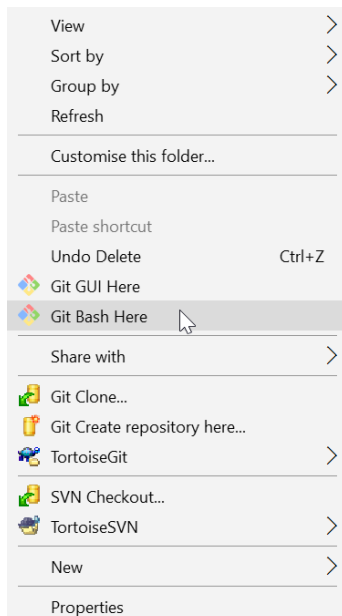
хранилище. После натиснете OK.

- Резултатът би трябвало да е нещо подобно на:





- Използвайте **"git clone"** командата за клониране с **GitBash**.
 - Отидете в желаната **папка** (различна от първата), щракнете на десен бутон някъде в празното пространство на папката, изберете **"Git Bash here"** напишете командата **"git clone"**, следвана от URL адреса на вашето хранилище.



- Резултатът трябва да е нещо такова:

```
MINGW64: ~/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: ~/Desktop/Repositories/GitBash
$
```

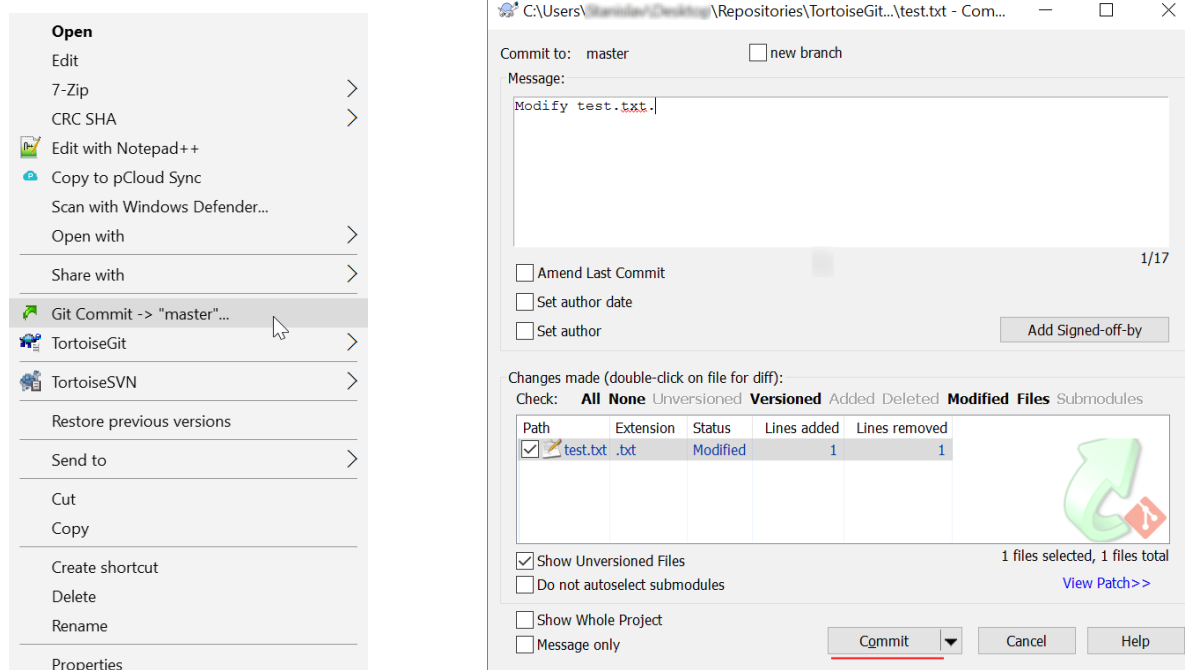
4. Създайте конфликт

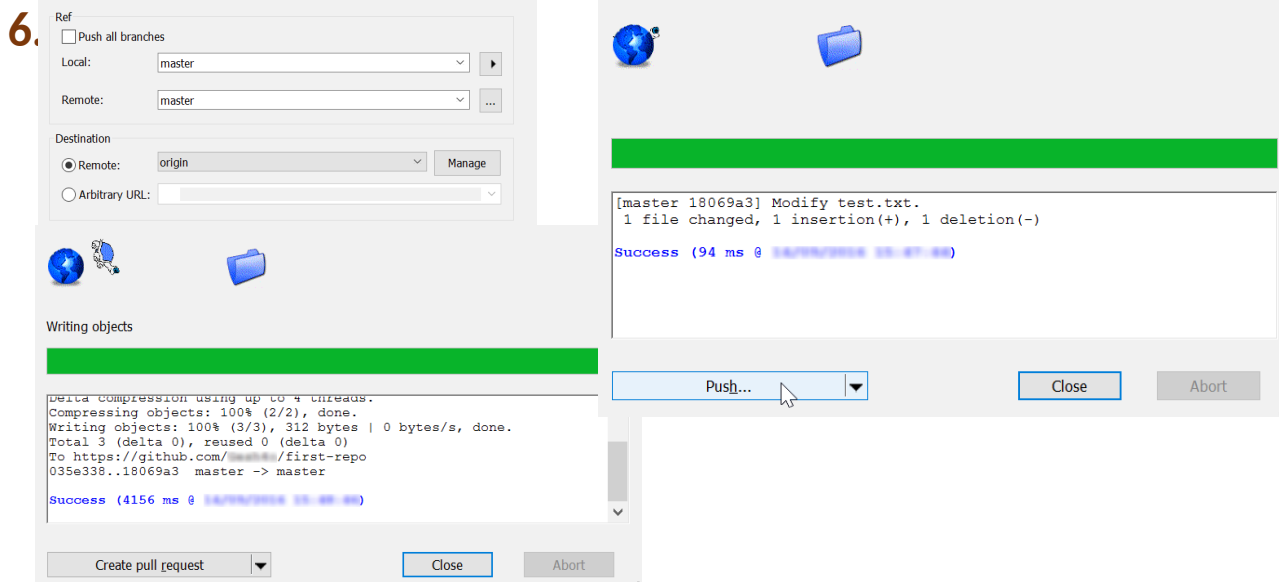
Променете съдържанието на двете папки по различен начин:

- В клонираното чрез **TortoiseGit** отворете файла **test.txt** и добавете ред: "Промяна с Tortoise..."
- В клонираното чрез **GitBash** отворете файла **test.txt** и добавете ред: "Промяна с Bash..."

5. Commit-нете промените от TortoiseGit копието.

- Може да използвате "Git Commit..." командата на TortoiseGit. Напишете кратък текст в полето Message, описващ каква е промяната:





Опитайте сега да обновите вашето GitBash копие.

- Отворете другата ви папка с копие на Git и в нея извикайте **GitBash** конзолата. Изпълнете следните команди:
 - **Покажете** всички променени файлове
 - "git add ."
 - **Commit-нете** вашите промени, заедно с обяснение за промените в commit-a.
 - "git commit -m "Update test.txt.""
 - **Обновете** локалното ви хранилище
 - "git pull"

```
MINGW64/c:/Users/.../Desktop/Repositories/GitBash/first-repo
@DESKTOP-FNOG4KM MINGW64 ~/Desktop/Repositories/GitBash/first-repo (master)
$ git add .
@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(-)
@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/.../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.
@DESKTOP-FNOG4KM MINGW64 ~/Desktop/Repositories/GitBash/first-repo (master|MERGING)
$
```

7. Сега имаме конфликт при сливането, който трябва да разрешим.

- Отворете файла **test.txt** във вашето **GitBash** копие, трябва да е нещо подобно на:

```
test.txt - Notepad
File Edit Format View Help
<<<<<<< HEAD
Updating with Bash...
=====
Update with Tortoise...
>>>>>>> fdb74be0d6e6dc9de9a4a5229da7c4277f1e0066
```

- Премахнете **HEAD**, **=====**, **<<<<<<**, **>>>>>>** символите и запишете файла.

```
test.txt - Notepad
File Edit Format View Help
Updating with Bash...
Update with Tortoise...
```

- Сега, след като сте разрешили **конфликта**, покажете променения файл (с **git add .**), **commit-нете** пак промените (с **git commit -m "съобщение"**) и **синхронизирайте** вашите промени с отдалеченото хранилище (с **git sync**).

```
MINGW64: c:/.../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.

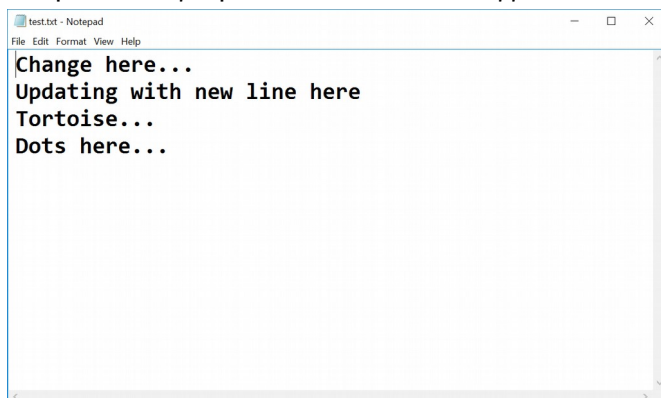
$ git add .
$ git commit -m "Merge commit."
[master 1c353f7] Merge commit.

$ 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
```

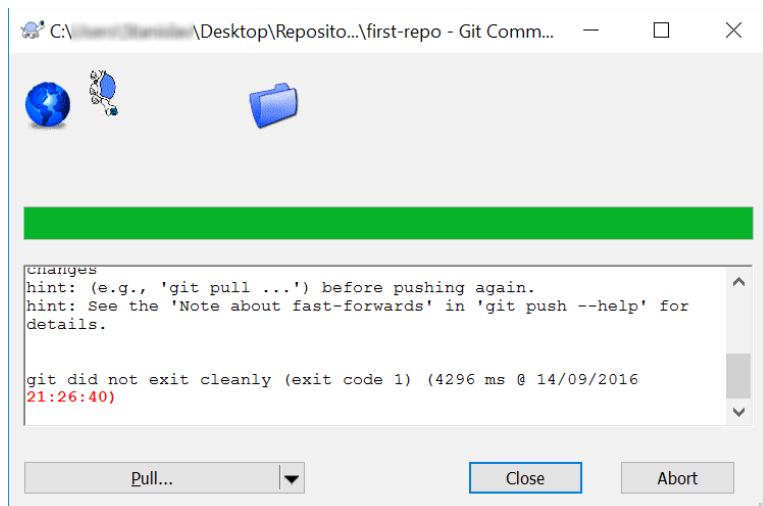
8. След като обновихте съдържанието на отдалеченото хранилище, сега опитайте да обновите вашето TortoiseGit копие.

- Направете още промени в test.txt и им дайте **commit**.

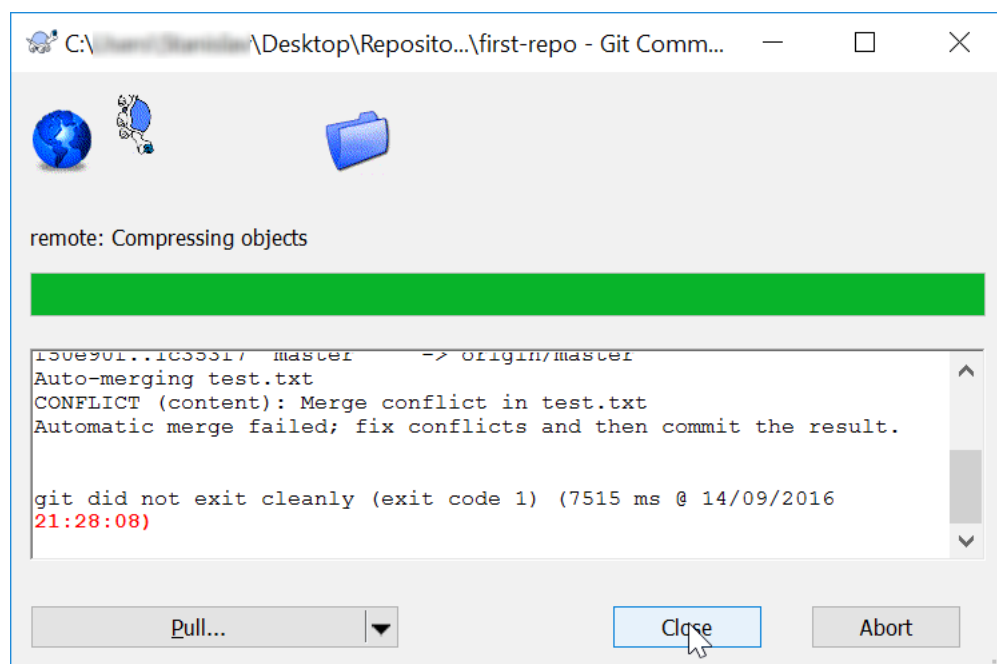


* **Бележка:** ако промените са прекалено прости, TortoiseGit **автоматично** ще ги слее.

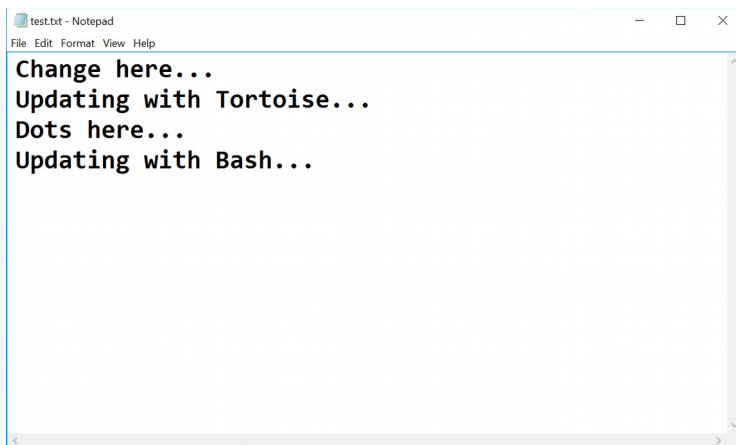
- Опитайте сега да ги **изпратите** (първо на десен бутон с „Git Commit -> “master”” и после с бутона [Push]) – не става, нали? Оказва се, че нашето **отдалечено** хранилище е **обновено** (със commit-а, който направихме след сливането) и тези промени не присъстват в **локалното** ни хранилище.



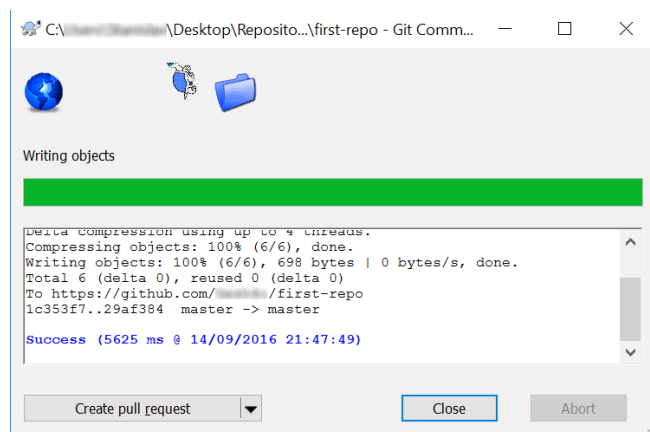
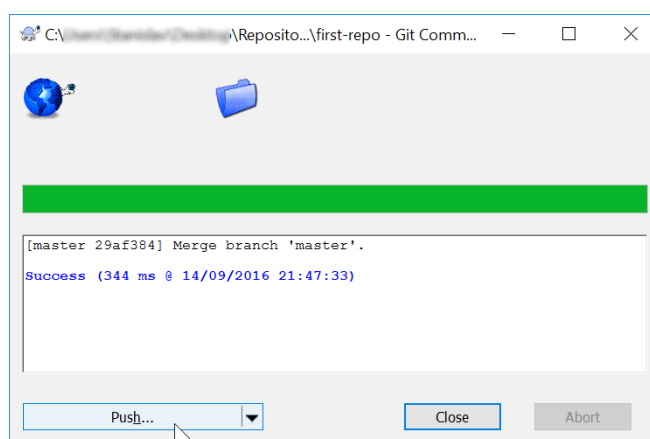
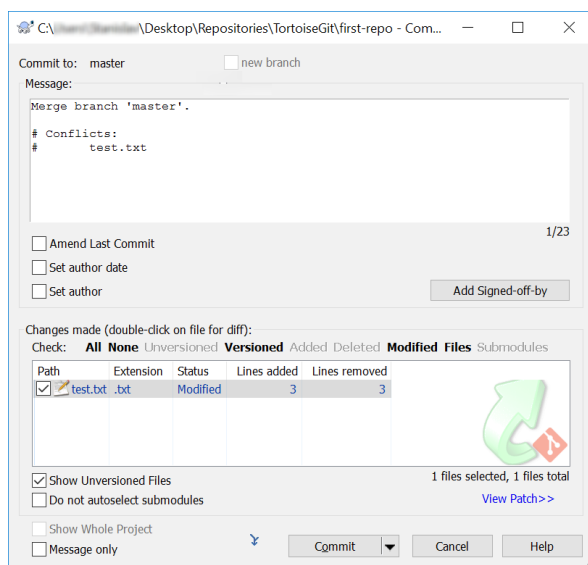
- Така че трябва да **изтеглим** (с **pull**) новите промени:



- Обърнете внимание на съобщението: "Automatic merge failed; fix conflicts...". Имаме нов конфликт и ще трябва да го разрешим както преди, но с малка разлика:
 - Отворете **test.txt** файла, **премахнете** същите **символи** както предния път. После щракнете с десен бутон върху файла и изберете **TortoiseGit -> Resolve...** Ще се отвори диалогов прозорец, там цъкнете на "Ok", за да се опитате да **разрешите** конфликта.



- Сега нашият файл е **чист** и сме готови за нашия окончателен **commit**!

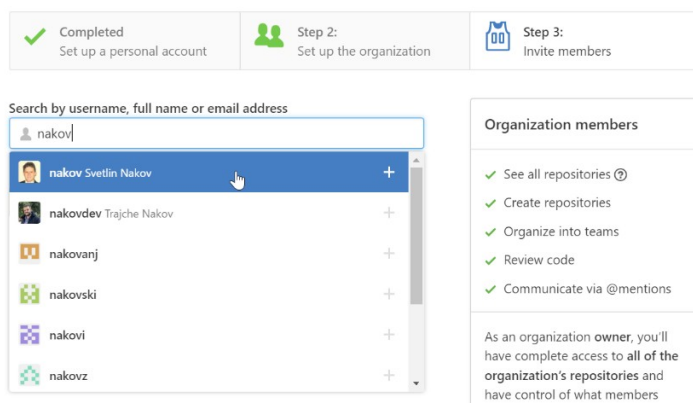


III. Екипна работа

Работа в **екип** от (около) 5 ученика в клас. Всеки екип трябва да си избере „**ръководител на екипа**“. Ръководителят **създава организацията** в GitHub:

- Нова организация от: <https://github.com/settings/profile>, страница: *Organizations*.
- Избира **уникално** име за организацията, например "**SoftUniOrg**" и добавя членове в нея.

Invite organization members



- След това създава хранилище, например "test-repo"

9. Добавете файл към GitHub

Членовете на екипа (включително и ръководителя на екипа) добавят няколко файла:

1. Клонирате "test-repo" на вашия компютър.
2. Създайте нов файл във вашата работна директория.
 - Наречете файла <вашето_име>.txt
3. Добавете някакъв текст в него, например „Казвам се“
4. Commit-нете **новия файл** във вашето **локално хранилище**.
5. **Синхронизирайте** промените за да **изпратите файла в отдалеченото хранилище**.
6. Разгледайте хранилището на адрес <https://github.com/user/repo>, за да проверите дали файлът ви е успешно изпратен в GitHub.

10. Създайте конфликт в Git и слейте промените

- Нека всички членове на екипа създадат общ файл **config.txt**
- Всеки член на екипа да добави някакви настройки във файла **config.txt**, например
 - *Име = Петър*
 - *размер = 100*
 - *email = peter@dir.bg*
- Всеки член на екипа **commit-ва** своите локални промени.
- Всеки член на екипа **синхронизира** своите промени с общото хранилище.
 - При първия това ще стане успешно, без **конфликти**.
 - Другите ще имат конфликт, промените от който трябва да бъдат **слети**.
 - **Разрешете** конфликта:
 - **Редактирайте** слетите промени + **commit** и **синхронизирайте** отново.