# Упражнения: Git и GitHub

Задачи за упражнения и домашно към курса [“Основи на програмирането” @ СофтУни](https://softuni.bg/courses/programming-fundamentals).

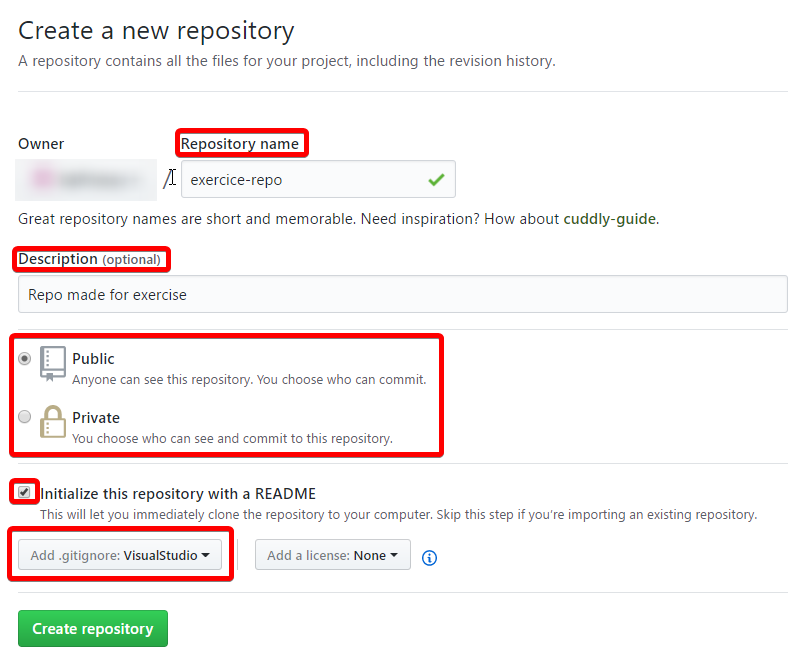
# TortoiseGit

## Качване на проекти в GitHub

Създайте няколко **хранилища** във вашия **GitHub** профил и **качете няколко от вашите проекти в GitHub**. Това може да са **упражнения за домашна работа** от последните часове, ваши **екипни** проекти или всякакви други проекти, които бихте искали да споделите с другите разработчици. Направете го в следните стъпки:

1. **Създайте отдалечено хранилище за вашия проект**

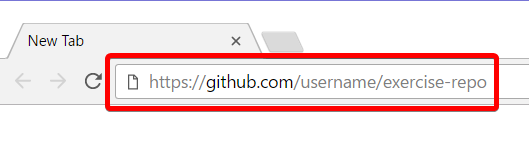
Отидете на адрес <https://github.com/>. Щракнете на бутона **New repository,** би трябвало да видите следния екран:



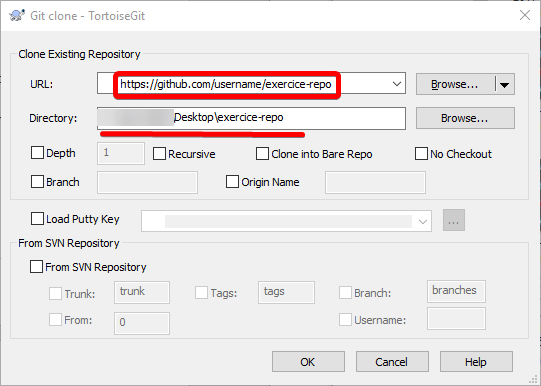
В полето “**Repository name**” може да напишете името на вашето ново хранилище. Може също да добавите описание (в полето **description**)и евентуално да промените **видимостта** (от полето **visibility**)на вашето хранилище. Добър стил на работа е да добавите **README** към вашето хранилище. Така може да добавите повече информация за вашия проект. Просто изберете отметката за създаване на **README** и GitHub ще създаде файла вместо вас.

1. **Клонирайте** го на вашето устройство:

Можете просто да копирате **URL**-а на вашето хранилище:



След това, поставете **URL**-а в **TortoiseGit** и той ще **клонира** хранилището локално на вашия компютър:

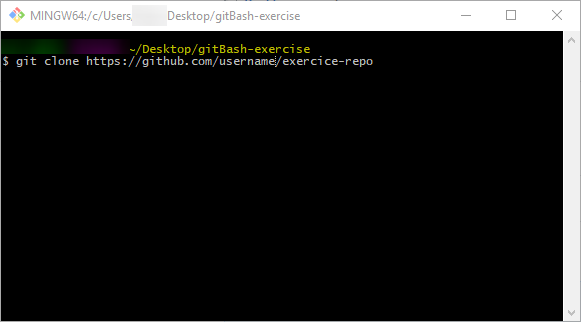


В примера по-горе хранилището е клонирано на работния плот (в папка **Desktop**), но вие може да го направите в друга папка по ваш избор.

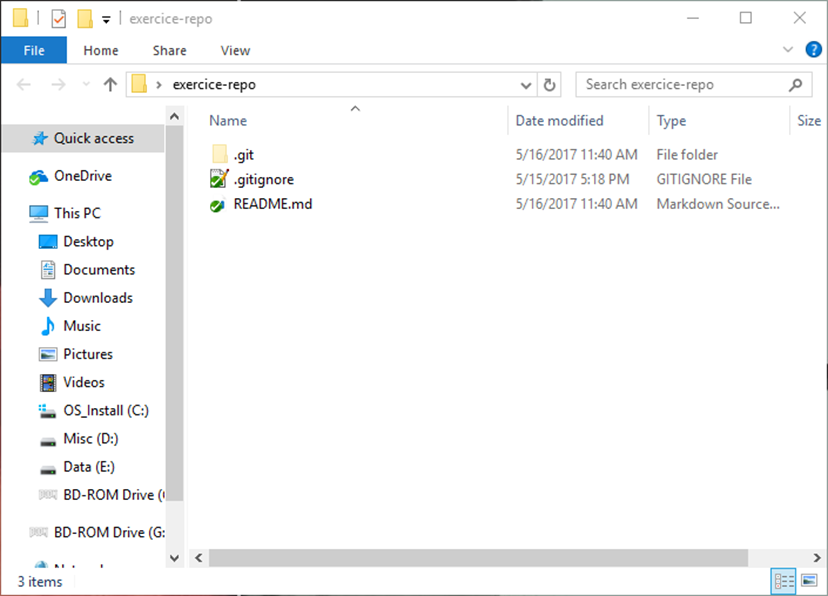
Обърнете внимание че **всички (безплатни) проекти** които качите в GitHub ще бъдат **с отворен код** и ще са достъпни за всеки в Интернет, така че внимавайте за пароли или **програмен код**, който **не** бихтеискали да бъде **видим** от някой **друг**. Ако искате, можете да прочетете повече за договорите за ползване [тук](https://choosealicense.com/).

**Клонирайте** някои от вашите GitHub хранилища чрез **Git клиента**, който ползвате (примерно **TortoiseGit** или **GitBash**). Направете някакви локални **промени**, след това ги **commit-нете** и **изпратете** (с **push**) към GitHub. Проверете дали промените са публикувани в GitHub профила ви в Интернет. Стъпките да го направите са:

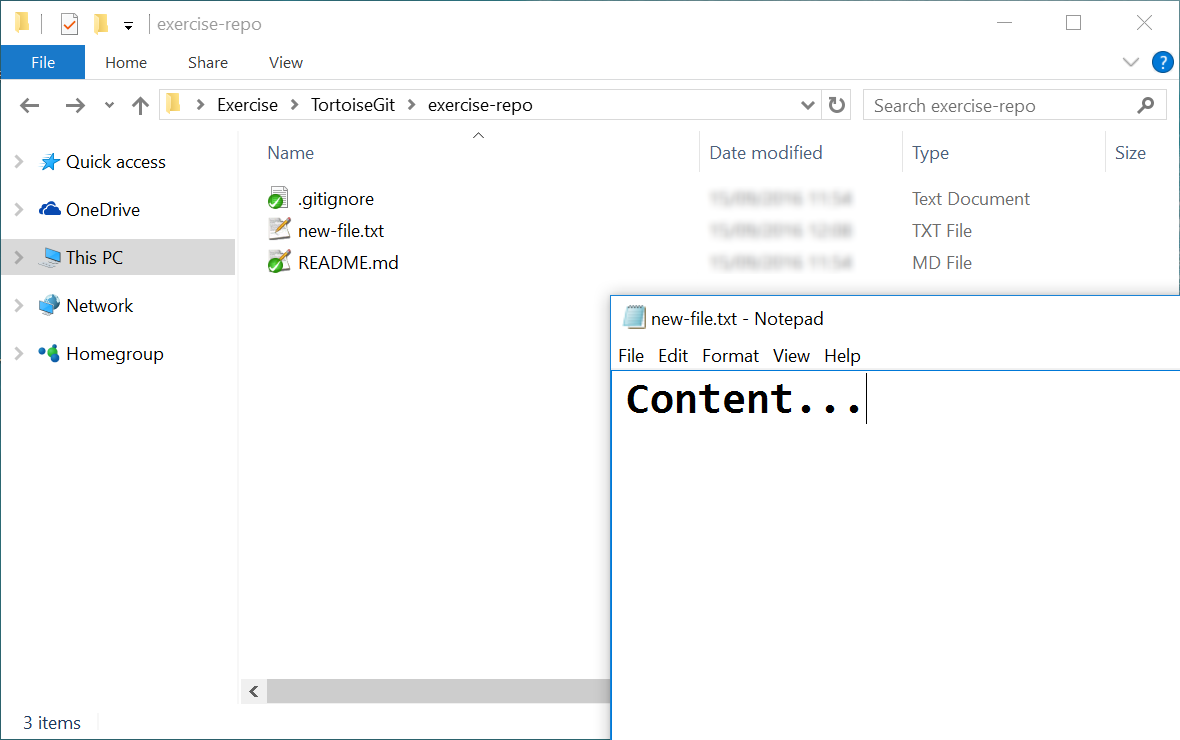
1. **Клонирайте** хранилището пак, но в друга папка (този път използвайте **GitBash**, с командата*"*git clone*"*):



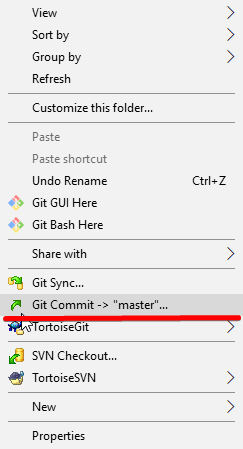
1. Върнете се на предишното копие на хранилището и **го отворете** в **Windows Explorer**. Добавете нов файл в папката:



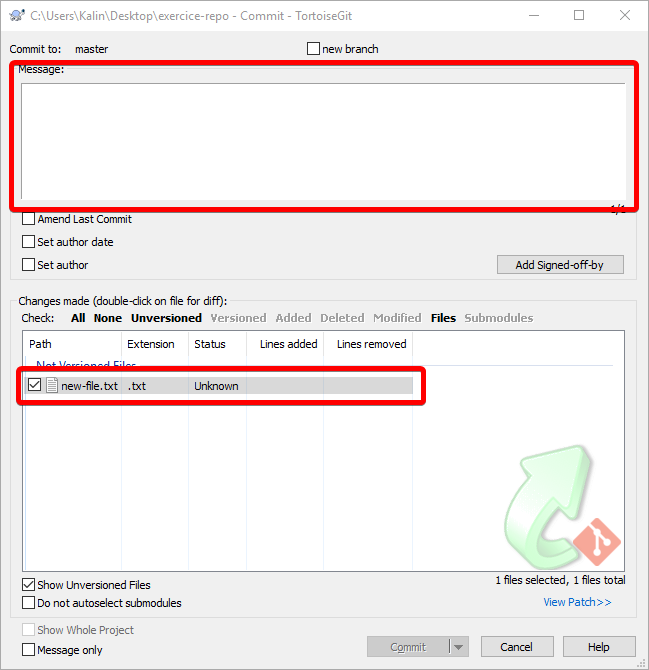
1. Направете някакви **промени** във файла new-file.txt:

****

1. **Commit**-нете вашите локални промени към локалното ви хранилище. Щракнете **с десния бутон** и после от менюто на командата „**Git Commit**”

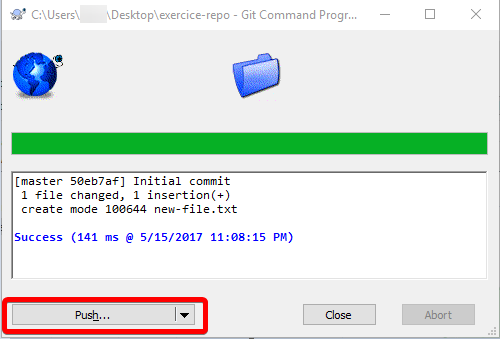


Ще видите следния прозорец:



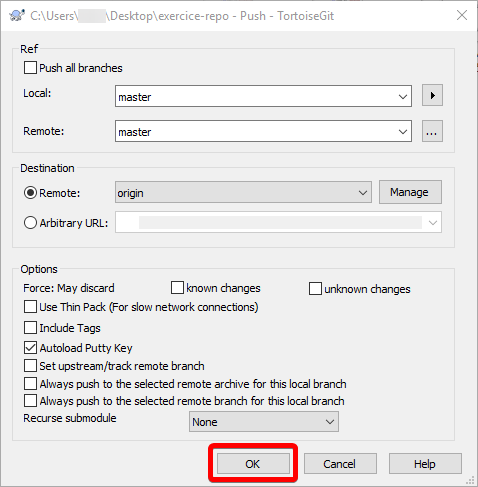
В секцията за съобщения, напишете **кратко обобщение** на промените във вашия commit. Добра практика е обясненията да са смислени. **Не пропускайте** да добавите и файловете си в **долната** част на прозореца.

Когато сте готови с тези стъпки, можете да натиснете **[Commit]** и би трябвало да видите **следния прозорец**:



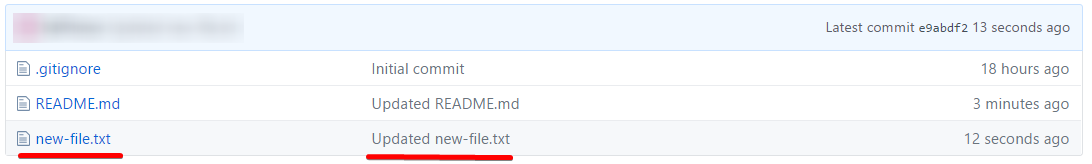
В него се вижда **колко** файла са **променени** и **колко вмъквания** и/или **изтривания** са направени. След като се запознаете с информацията, натиснете бутона **[Push]**.

1. **Изпратете** (**push**) вашите промени към отдалеченото хранилище в GitHub:

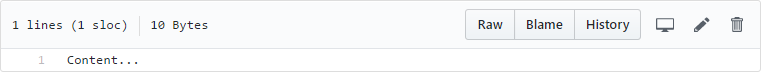


От този прозорец управлявате в кое **разклонение** ще бъдат изпратени вашите файлове, но за разклоненията ще говорим по-нататък в това упражнение. За момента, просто натиснете **[OK]** и файлът ви ще бъде **изпратен** в **разклонението**, наречено **master**.

1. Проверете дали вашите промени са видими **онлайн**:



Отворете **вашето GitHub хранилище** в браузъра и цъкнете на **new-file.txt**. В него би трябвало да видите съдържанието, което сте добавили. На горната снимка на екрана се вижда, че **commit** **съобщението** на commit-a,който сте направили, е записано във **втората колонка**. Можете да използвате тази колона, за да **получите повече информация** за това кои файлове са променени и каква точно е промяната. **По тази причина винаги е добра практика да пишете достатъчно обяснителни commit съобщения**. След като щракнете на файла, би трябвало да видите нещо подобно:

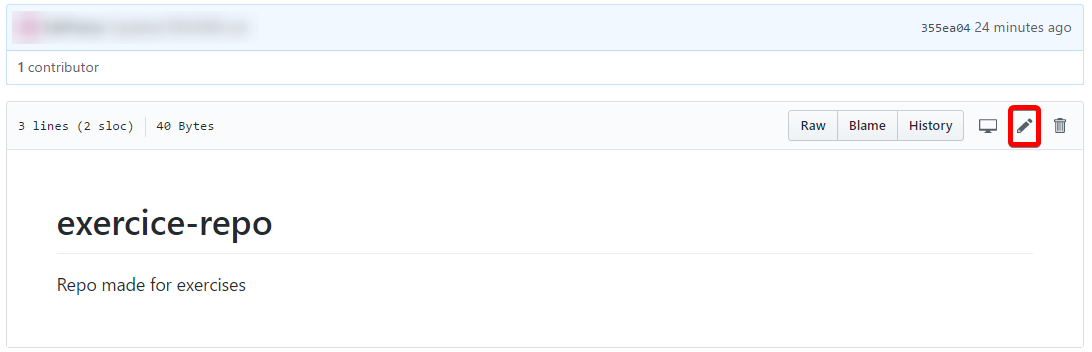


## Създаване на конфликти и разрешаването им

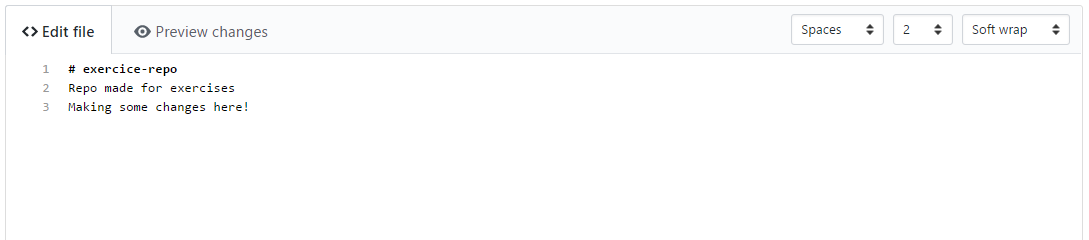
Може би сте забелязали, че във вашето хранилище присъства файл, наречен **README.md**. Той се използва за да напишете **ръководство**  за вашето **приложение** или просто за да дадете **повече информация**  за вашия проект. Този файл използва **маркиращ език**  наречен “[Markdown](https://en.wikipedia.org/wiki/Markdown)”. Този език се използва основно за **форматиране на текст** и за **писане на readme файлове**.

А сега нека направим **конфликт** в нашия **README.md**.

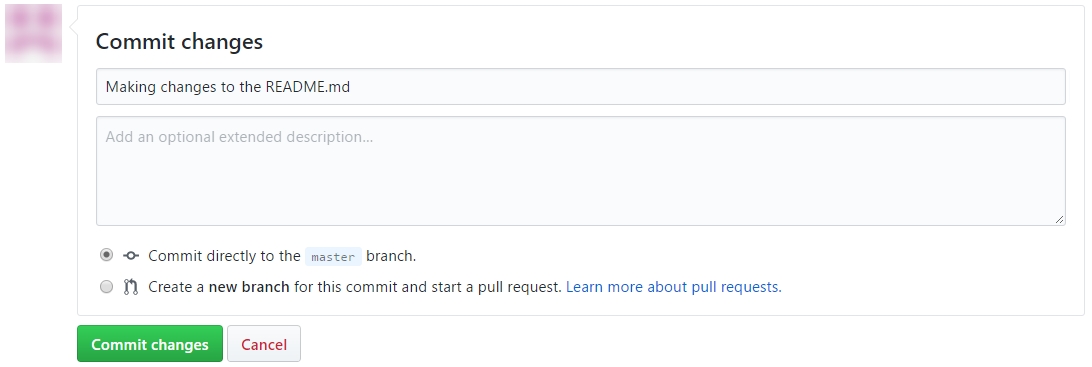
1. Отворете вашата GitHub регистрация във вашия **уеб браузър.** Цъкнете на **README.md** и после на **молива**  в **горния десен ъгъл**:



Ще видите **текстовия редактор на GitHub**:

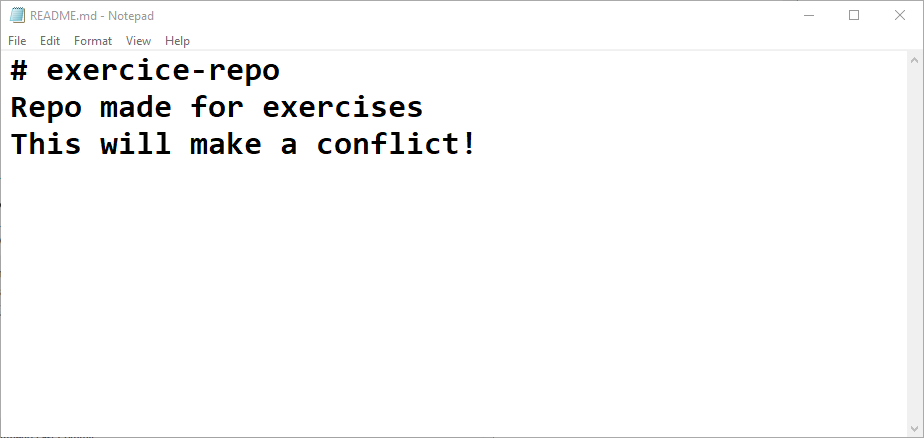


**Направете някакви промени** във файла и **скролирайте надолу**. В дъното на страницата ще видите следното:

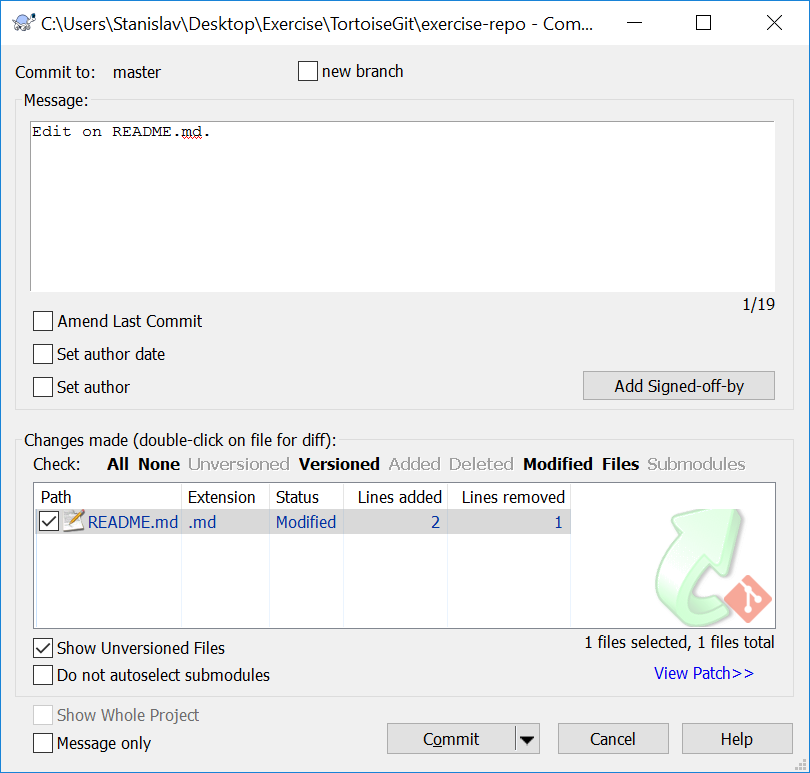


Тук можете да напишете вашето **commit** **съобщение**. След като сте готови, щракнете на **[Commit changes]**

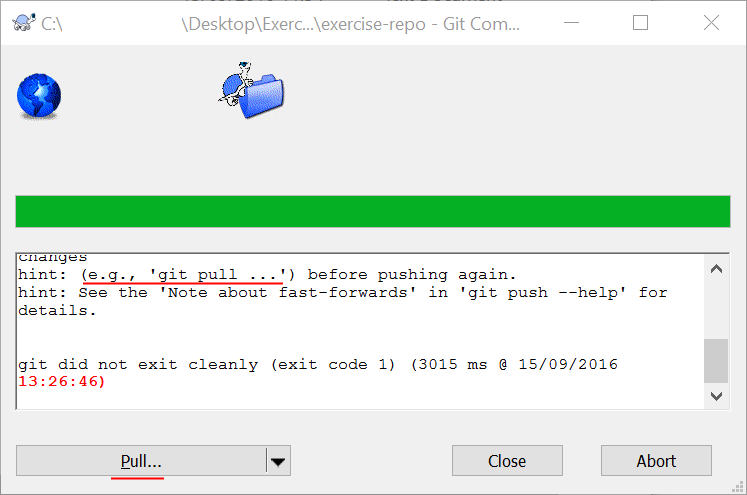
1. Отворете вашето локално копие на хранилището и в него – файлът **README.md** (**без да изтегляте с pull промените**). После добавете някакъв друг, различен от първия, текст:



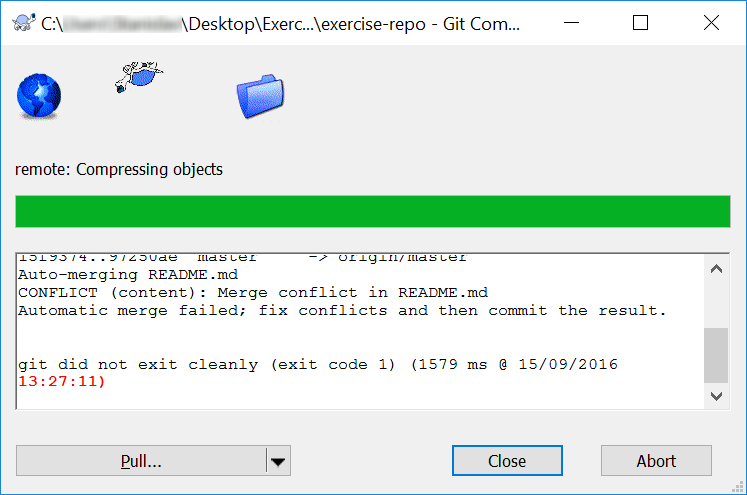
1. Сега направете **commit** на локалните ви промени с помощта на **TortoiseGit**:



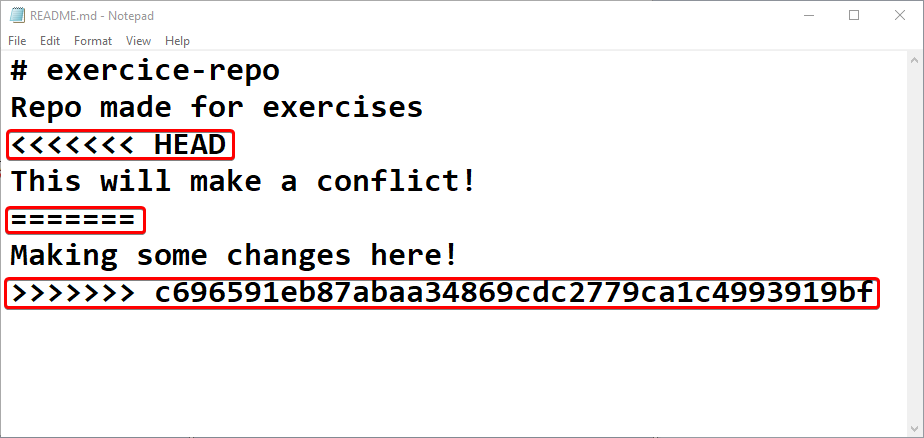
1. Опитайте се да **изпратите** (т.е. да направите **push** на) локалните промени към **отдалеченото хранилище.** Операцията ще бъде **неуспешна**, понеже отдалеченото хранилище е **обновено**, а локалното **не е**:



1. След изтегляне **TortoiseGit** **ще се опита** да изтегли (pull) и слее (merge) промените (без успех), така че ще трябва да направим сливането **на ръка**.



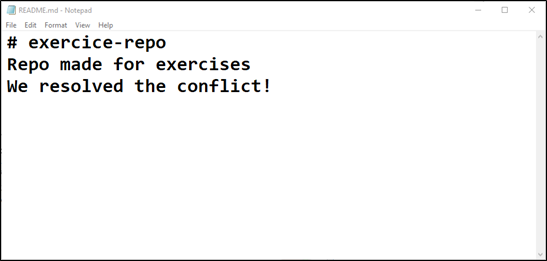
1. Сега **разрешете конфликта**:



(**<<<<<<< HEAD**)маркира **началото на локалната версия на файла**; (**========**) **разделя локалната версия от тази в хранилището**. (**>>>>>>>**) **маркира края на файла** и след него е записан номера на **commit**-а. За да разрешите конфликта, трябва да изтриете всичките три специални маркера и да изберете коя версия на файла да запазите. Имате три възможности:

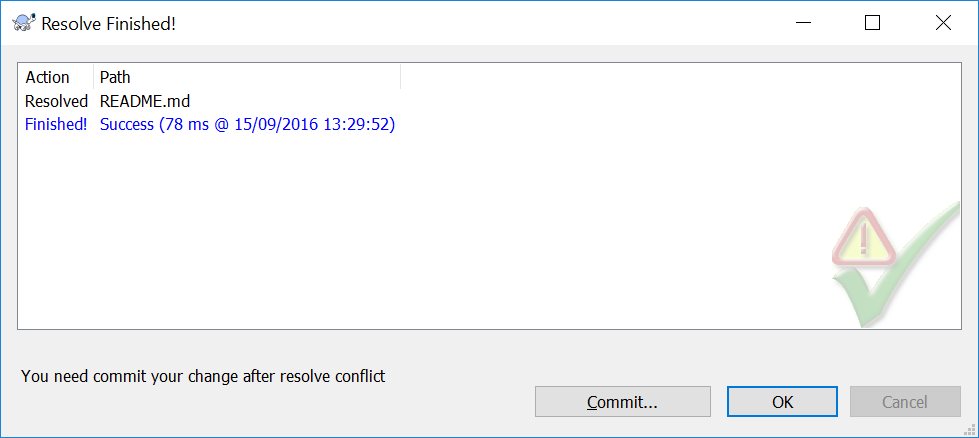
* Можете да **изтриете** “**This will make a conflict!**” или “**Making some changes here!**” (т.е. единия от двата различни текста, породили конфликта);
* Можете **да запазите и двата текста** и **да изтриете само маркерите**;
* Или можете **да напишете** напълно различно, **ново изречение**. ☺

В тази екранна снимка е избран **третия** подход и пишем **нов текст**:

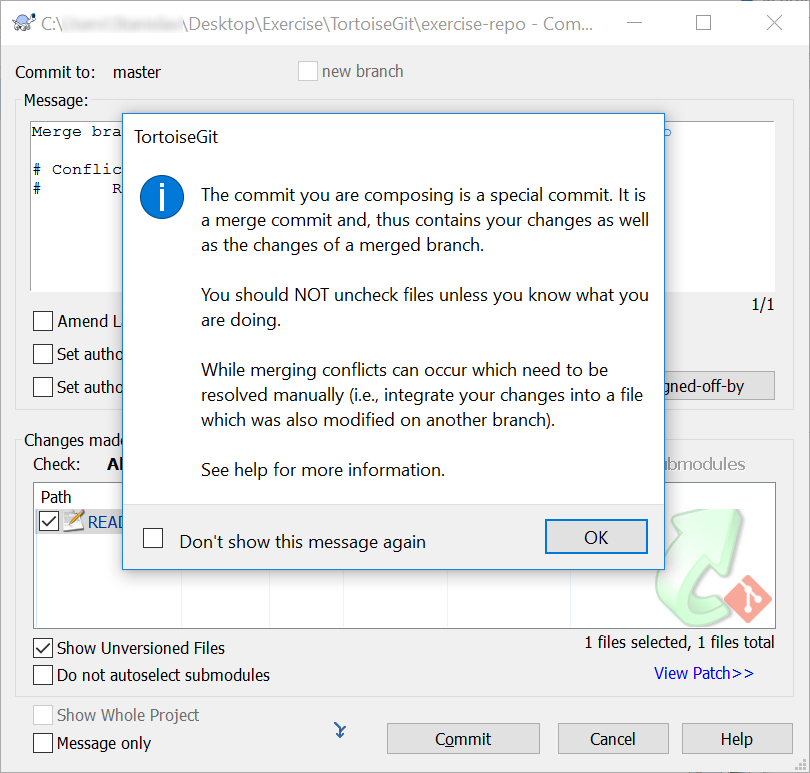


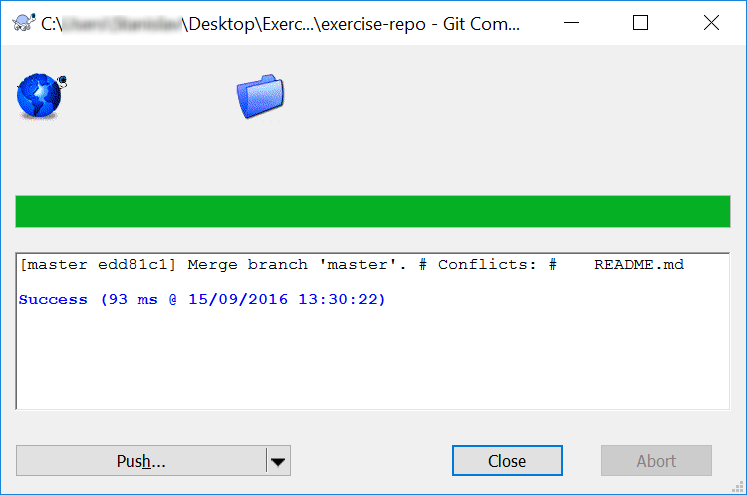
В тези примери използваме **Notepad** за редактиране на файловете, но повечето **интегрирани среди** (**IDE:** Visual Studio, Eclipse, IntelliJ, WebStorm и други)имат **Git интеграция** и ще ви покажат **разликите, които пораждат конфликт**.

1. Обявете текущия файл за такъв с разрешени конфликти от **TortoiseGit** -> **Resolve**

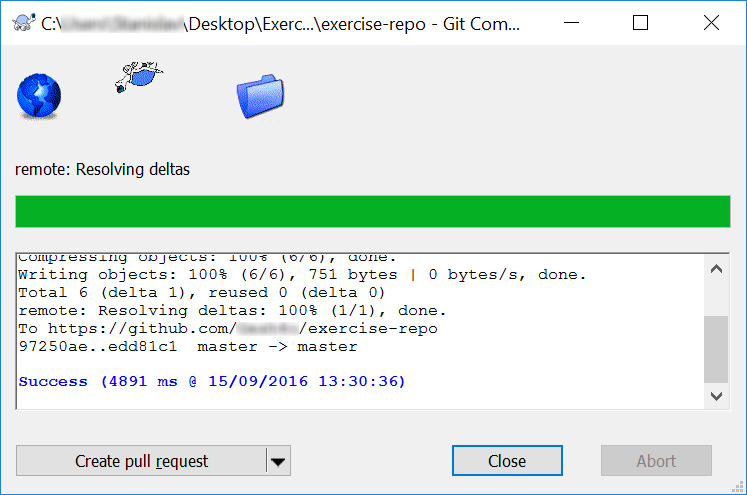


1. **Commit-нете слетите промени** (направените локално и тези, които направихте през уеб браузъра):

****

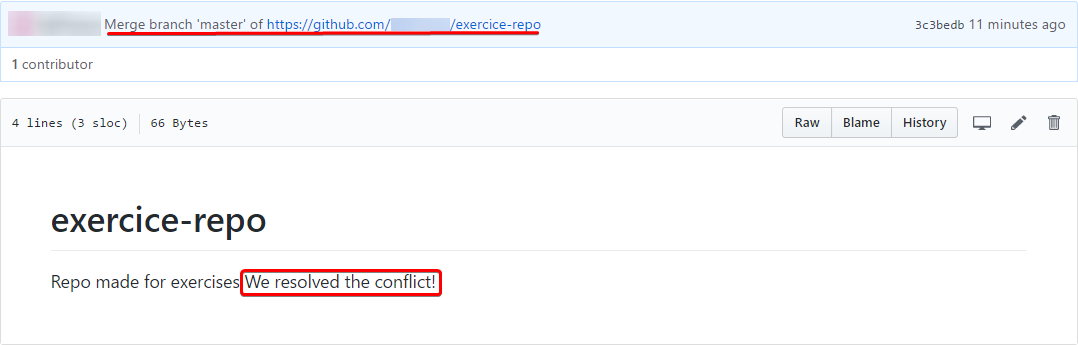


1. Сега **изпратете отново** (с **push**) вашитепромени към онлайн хранилището в GitHub.



Супер, **commit-a би трябвало да е успешен** и **без конфликти!**

1. Накрая, **проверете** качени ли са **промените** в уеб през вашата GitHub регистрация:



Обърнете внимание, че когато се прави merge, **commit**-а е със **специално описание**.

## Създаване на разклонение и сливане на промени

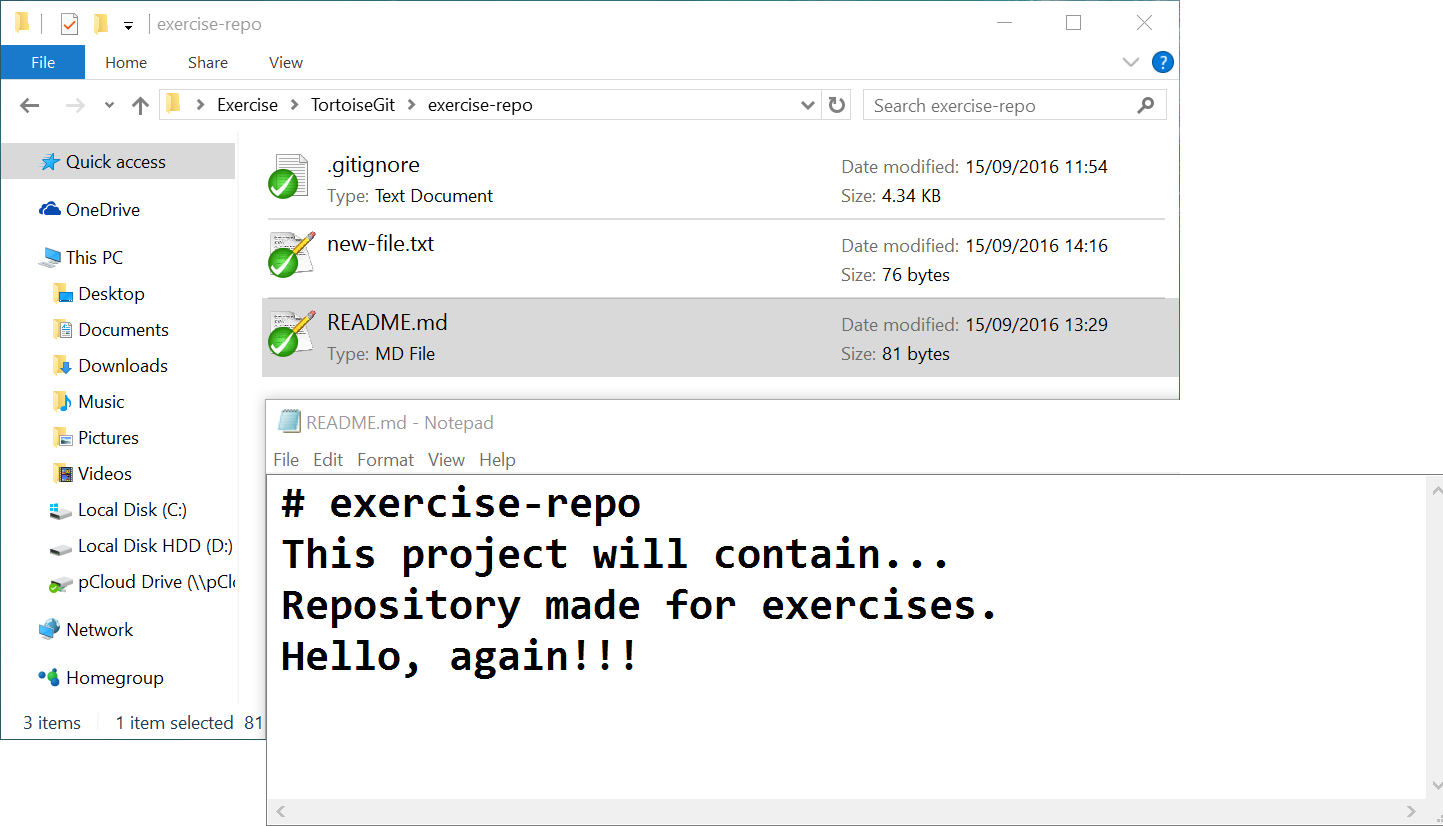
Разклоненията (branches) са много **полезни когато много хора работят по един проект.** Такива случаи са **предпоставка за голямо количество конфликти**. С разклоненията разработчиците имат възможността да работят върху **отделни части от проекта** без да предизвикват конфликти. Когато някой от програмистите завърши новите функционалности върху които работи, **разклонението се слива обратно с основното (main) разклонение на проекта**.

1. Създайте **разклонение.** (В този пример името му е: **develop**)

|  |  |  |
| --- | --- | --- |
|  | 🡺 |  |

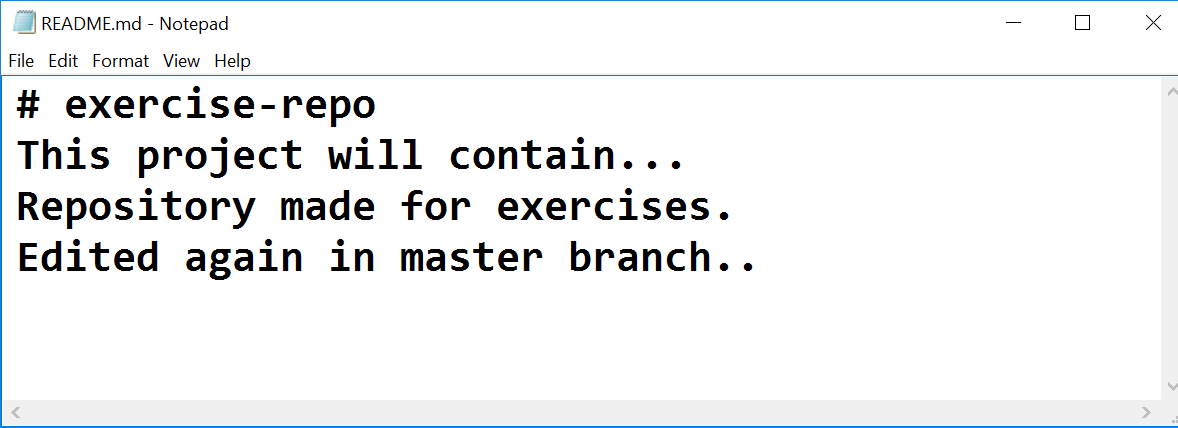
1. **Превключете** към това разклонение.   
   

|  |  |  |
| --- | --- | --- |
|  | 🡺 |  |

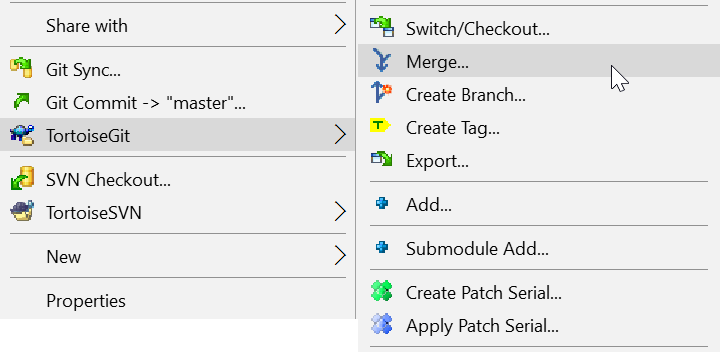
1. Направете някакви **промени. Редактирайте** един от файловете във вашето хранилище.  
   ****
2. **Commit**-нете както преди.
3. **Превключете** към основното (main) разклонение.



1. Направете някакви промени в основното разклонение (върху същия файл, който редактирахте и преди малко). **Commit-нете промените** и после ги **изпратете** (с **push**).

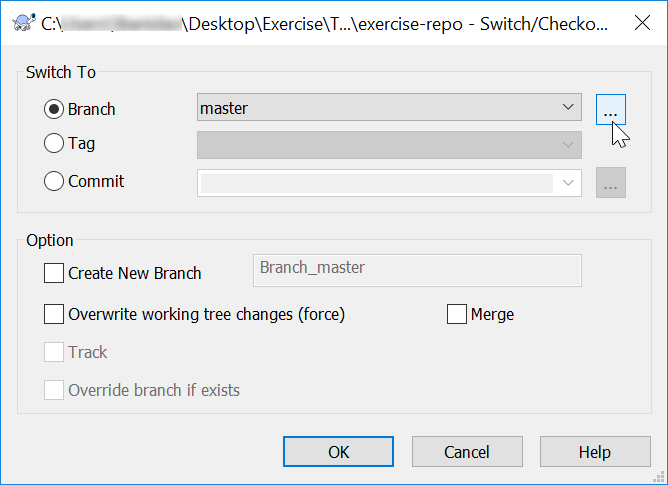


1. **Слейте** с промените от другото разклонение (в нашия случай - **develop**).

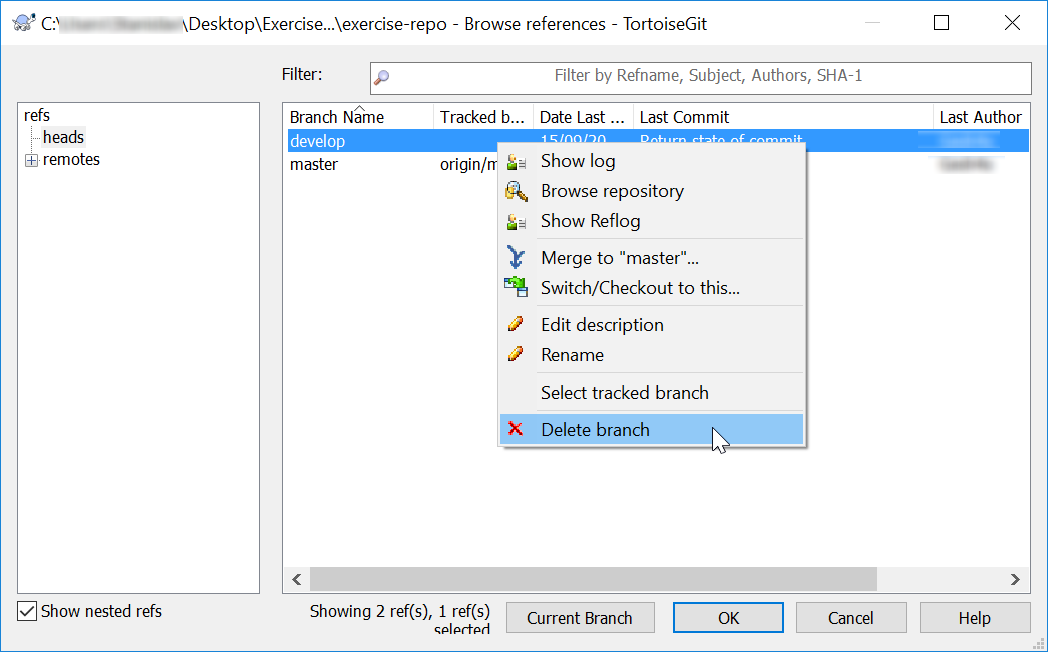


|  |  |  |
| --- | --- | --- |
|  | 🡺 |  |

1. **Разрешете** новите конфликти и commit-нете промените в хранилището.
2. **Изтрийте** новосъздаденото разклонение (**develop**).
   1. Изберете **TortoiseGit** -> **Switch/Checkout…**



* 1. Щракнете на десен бутон върху **избрания** елемент […] по-горе и би трябвало да се появи прозорец като този:



* 1. Оттук можете да **изтриете** разклонението и да **commit**-нетепромените си.

1. **Обновете (update)** отдалеченото хранилище.

# GitBash

**GitBash** е конзолен клиент за **GitHub**. Много програмисти го ползват, защото **предоставя повече контрол и изпълнява само командите, които сте написали**. Повечето графични клиенти като TortoiseGit **изпълняват фоново и допълнителни команди**, което може да е проблем в по-големи проекти.

## Изпращане на проекти в GitHub

**\*** Ако вече сте клонирали хранилището си с **GitBash** спокойно можете да пропуснете тази стъпка.

1. **Клонирайте** на вашето устройствосъщото хранилище, което ползвахте за предишните задачи:

* Използвайте командата "git clone".

1. Отворете файловете в **Windows Explorer**.
2. Направете някакви **промени** с любимия ви текстов редактор.
3. **Commit**-нетелокалните ви промени в локалното хранилище.

* Използвайте командата "git add". Може да напишете "git add ." като команда в **GitBash**. Тази команда **подготвя** (**stages**) всички **нови**  и **променени** файлове за записване.
* Завършете с командата "git commit".

1. **Изтеглете (pull)** и после **изпратете (push)** вашите промени към отдалеченото хранилище в GitHub:

* Първо с "git pull" изтеглете евентуално появилите се отдалечени промени.
* Използвайте "git push" за да изпратите локалните ви промени към отдалеченото хранилище.

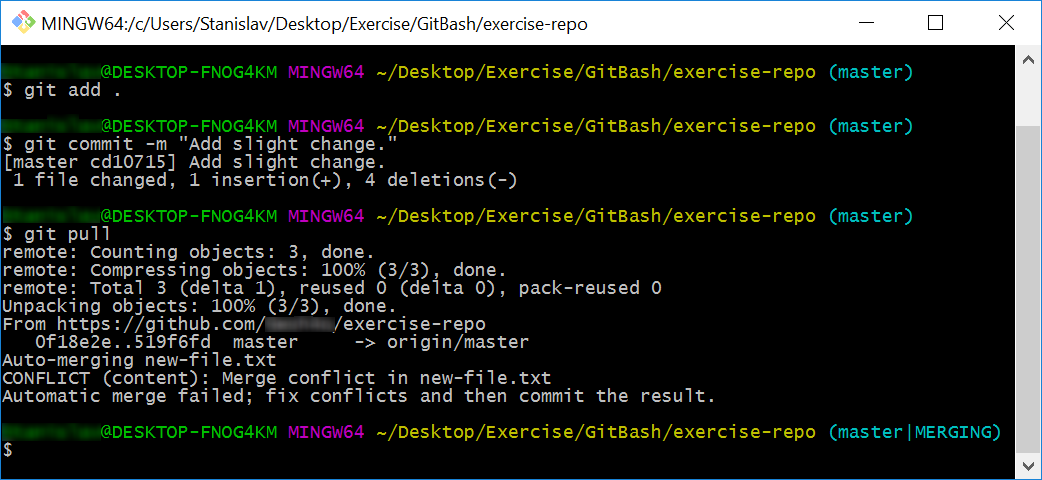
1. Проверете дали промените ви са на сървъра на GitHub.

## Създаване на конфликти и разрешаването им

Направете някакви **промени, пораждащи конфликт** и ги **слейте**. Стъпките са тези:

1. Извършете някаква **промяна** в локалната ви директория, например редактирайте файла README.md.
2. **Не commit**-вайте и **не push**-вайте все още промените си.
3. Отворете **GitHub** регистрацията си през вашия **уеб браузър** или през **TortoiseGit**. Направете някакви промени върху същия файл.
4. Сега ги **commit**-нете.
5. Опитайте се да **обновите** локалните ви файлове с промените от **отдалеченото хранилище** в GitHub:

* Използвайте командата "git pull".

1. Ще получите **уведомление за конфликт**.  
   

Един от файловете в **локалното хранилище** ще бъде **обединен**  с неговата по-нова версия от **отдалеченото хранилище**:

1. **Разрешете конфликта**. Редактирайте конфликтните файлове и помогнете те да бъдат коректно обединени. Изтрийте всички редове, които указват местата на конфликта при сливане (като <<<<<<< HEAD):
2. **Commit-нете слетите промени** (вашите локални промени и тези, направени през уеб/TortoiseGit):
3. **Синхронизирайте отново,** за да изпратите вашите промени към GitHub.

Сега, **обновяването би трябвало да е успешно** и **без конфликти**.

1. Накрая, **проверете какво е променено** през уеб, чрез GitHub регистрацията си или синхронизирайте вашето локално TortoiseGit хранилище.

## Създайте разклонение и слейте промените

1. Създайте **разклонение**.

* Използвайте командата "git branch branchName" за целта.

1. **Превключете** към това разклонение.

* Това става с командата "git checkout branchName".

**\* Бележка:** предните **2 стъпки** могат да обединени в една със **следната команда**:

"git checkout -b branchName"

1. Направете някакви **промени**.
2. **Commit**-нете вашите промени.
3. **Превключете** къмосновното (main) разклонение.

* *Вижте в стъпка 2 как става.*

1. Направете някакви промени в основното разклонение.
2. **Слейте** с предното разклонение.

* Това става с “git merge branchName”

1. **Разрешете** появилите се конфликти (ако има такива).

* Редактирайте файла за да разрешите конфликтите
* Накрая изпълнете “git add filename” и “git commit”

1. **Опитайте се отново да слеете (merge) - само** ако е имало **конфликти**  в стъпка 8).

* Използвайте “git merge branchName”

1. **Изтрийте** новосъздаденото разклонение.

* Използвайте командата “git branch -d branchName” за целта.

1. **Обновете** отдалеченото хранилище.
   * Използвайте командата “git push --all --prune”.