# Упражнения: Git и GitHub

Създаване на профил на разработчик в GitHub

## 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.



## 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.



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

## 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 ОК.

|  |  |
| --- | --- |
|  |  |

* 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:



## 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…*"*

## 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 .
* Предайте вашите промени, заедно с коментар за промените : git commit -m "Update test.txt."
* Обновете локалното ви хранилище: git pull

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

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



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



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

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

Graphical user interface, text, application, email

Description automatically generated

* 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-a, който направихме след сливането) и тези промени не присъстват в локалното ни хранилище.

Graphical user interface, application

Description automatically generated

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

Graphical user interface, text, application

Description automatically generated

* 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.

Graphical user interface, text, application, email

Description automatically generated

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

|  |  |
| --- | --- |
|  |  |
|  |

## 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 "**TestOrg**" и добавя членове в нея.

Graphical user interface, application

Description automatically generated

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

## Добавете файл към GitHub

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

1. Клонирайте "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.

## 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 синхронизиране отново.