

## Rozwiązywanie konfliktów

Julia Czarnowska

Karol Pisarski

Patryk Jar

29/11/2019



## Zadanie 1

- 1. Utwórz nowy folder na pulpicie o nazwie Lab3Gr[nr]
- 2. Zaloguj się na swoje konto na GitLabie
- 3. Wybierz z listy projektów projekt UGClass / Lab3 / Rz[nr]
- 4. Sklonuj repozytorium do utworzonego w poleceniu 1. folderu, przy użyciu adresu HTTPS (url)
- 5. Przejdź do lokalizacji: lab3/Gr[nr]/Zad1
- 6. Stwórz nowy branch zad\_1\_[imie]
- 7. Utwórz plik tekstowy ze swoim imieniem w nazwie (np. echo "" > Julia.txt)

#### Ściągawka

- \$ git clone <url> stworzenie kopii istniejącego repozytorium
- \$ git checkout -b <br/>branch> utworzenie i przełączenie się na nowego brancha
- \$ cd <file\_name> przejście do katalogu
- \$ Is polecenie wyświetla pliki w katalogu
- \$ mkdir <file\_name> utworzenie nowego katalogu
- klawisz Tab autouzupełnianie komend



## Zadanie 1 c.d.

- 1. Utwórz nowy commit (pamiętaj o poleceniu git add)
- 2. Przełącz się na mastera i uaktualnij repozytorium poleceniem git pull
- 3. Zmerguj swój branch do mastera
- 4. Wypchnij swoje zmiany do repozytorium (git push)

#### Ściągawka

- \$ git add dodanie pliku/plików do staging area
- \$ git commit -m "message" zapisanie commita z komentarzem
- \$ git checkout <brach> przełączenie się na istniejącego brancha
- \$ git merge <branch\_name> scalanie danego brancha z masterem
- \$ git push <remote\_repository\_name> <branch\_name> wysłanie wykonanych lokalnie zmian do zdalnego repozytorium







Konflikt powstaje gdy ten sam plik został zmieniony w różny sposób w tym samym miejscu w obu scalanych ze sobą branchach.

Może powstać podczas wywoływania różnych poleceń.



# Konflikty polecenie: git pull

Git wskazuje nam plik, który został zmieniony w dwóch miejscach. To w nim pojawił się konflikt, który należy naprawić.

```
[Satyr:ug_test_p1 karolp$ git pull
Auto-merging equation.txt
CONFLICT (content): Merge conflict in equation.txt
Automatic merge failed; fix conflicts and then commit the result.
```



# Konflikty polecenie: git diff

git diff <nazwa\_pliku> – wyświetlenie zmian będących w konflikcie

```
[Satyr:ug_test_p1 karolp$ git diff equation.txt
diff --cc equation.txt
index 3f8ae5b,1c81462..0000000
--- a/equation.txt
+++ b/equation.txt
@@@ -1,1 -1,1 +1,5 @@@
++<<<<<< HEAD
+f(y)=a*x-b
++======
+ f(y)=2a*x+b
++>>>>> 9847d29d2d26cfa264b8add28ece054211efaf9d
```



# Konflikty polecenie: git blame

git blame <nazwa pliku> – wyświetlenie zmian oraz ich autorów

Polecenie umożliwia nam sprawdzenie, kto wprowadził zmianę, z którą weszliśmy w konflikt. Możemy skonsultować, który pomysł był lepszy ©



## Zadanie 2

- 1. Stwórz nowy branch zad\_2\_[imie] i przełącz się na niego.
- 2. W istniejącym pliku "Zad2/zadanie\_2.txt" edytuj równanie "y=a\*x+b" w taki sposób aby za "a" podstawić nr rzędu a za "b" nr Twojego miejsca licząc od okna.
- 3. Dodaj zmiany i zrób commit w Twoim branchu.
- 4. Ściągnij aktualną wersję mastera i zmerguj Twój branch do mastera.
- 5. Wyślij zmiany do zdalnego repozytorium.
- 6. W przypadku konfliktów współpracuj z resztą zespołu aby je rozwiązać.

#### **ŚCIĄGAWKA**:

- \$ git checkout -b <br/>branch> utworzenie i przełączenie się na nowego brancha
- \$ git commit -m <message> zapisanie commita
- \$ git pull -- ściągnięcie aktualnej wersji ze zdalnego repo
- \$ git push wypchnięcie zmian do zdalnego repozytorium
- \$ git diff <file\_name> wyświetlenie zmian będących w konflikcie
- \$ git blame <file\_name> wyświetlenie zmian oraz ich autorów



# Konflikty polecenie: git mergetool

git mergetool – polecenie otwiera edytor graficzny umożliwiający porównanie obu wersji plików (zdalnej i lokalnej) i wybranie odpowiedniej

git config --global merge.tool <nazwa\_edytora> - konfiguracja mergetoola



## Zadanie 3

- 1. Upewnij się, że jests na **master**ze.
- 2. Stwórz nowego brancha.
- 3. Edytuj plik tekstowy z katalogu `zad3`. Add, commit, push.
- 4. Idź do gitlab.com i stwórz merge request.
- 5. Upewnij się, że koleżanki i koledzy z Twojego zespołu przejrzą zmiany i dodadzą swoje komentarze.
- Zmerge'uj swojego brancha do mastera (tak, napierw sam rozwiąż konflikty).

#### Ściągawka

- \$ git checkout master przejście na branch: master
- \$ git checkout -b lab3/zad3/krzysztof-jarzyna-ze-szczecina
- \$ git add.
- \$ git commit -m 'Moja zmiana'
- \$ git push



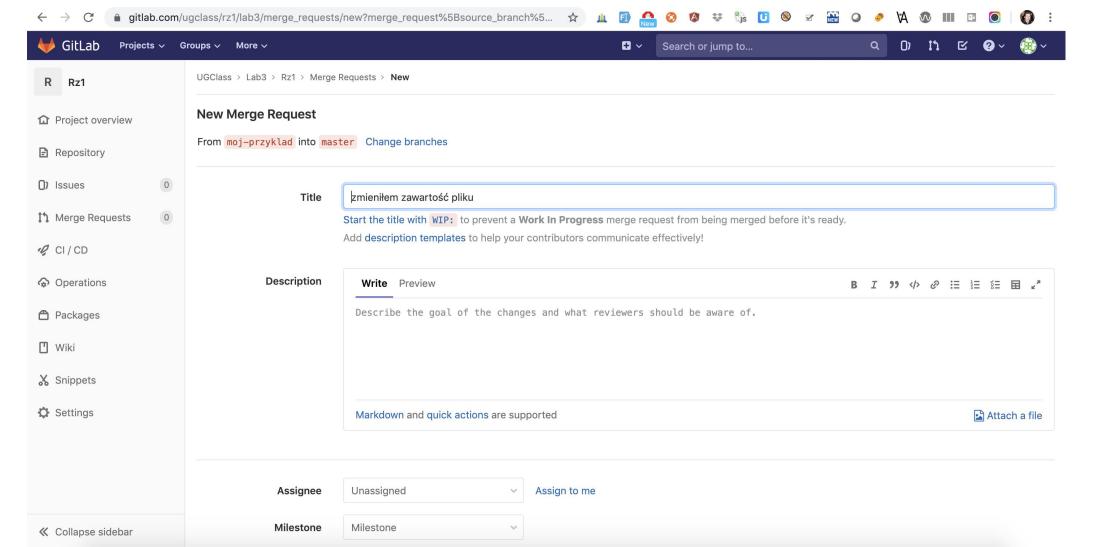
### Edycja pliku na branchu

### add, commit, push (--set-upstream)

```
[americano:lab3 patrykj$ git add .
[americano:lab3 patrykj$ git commit -m "zmieniłem zawartość pliku"
[moj-przyklad 5cbc5cc] zmieniłem zawartość pliku
1 file changed, 3 insertions(+), 3 deletions(-)
[americano:lab3 patrykj$ git push
fatal: The current branch moj-przyklad has no upstream branch.
To push the current branch and set the remote as upstream, use
git push --set-upstream origin moj-przyklad
americano:lab3 patrykj$ ■
```

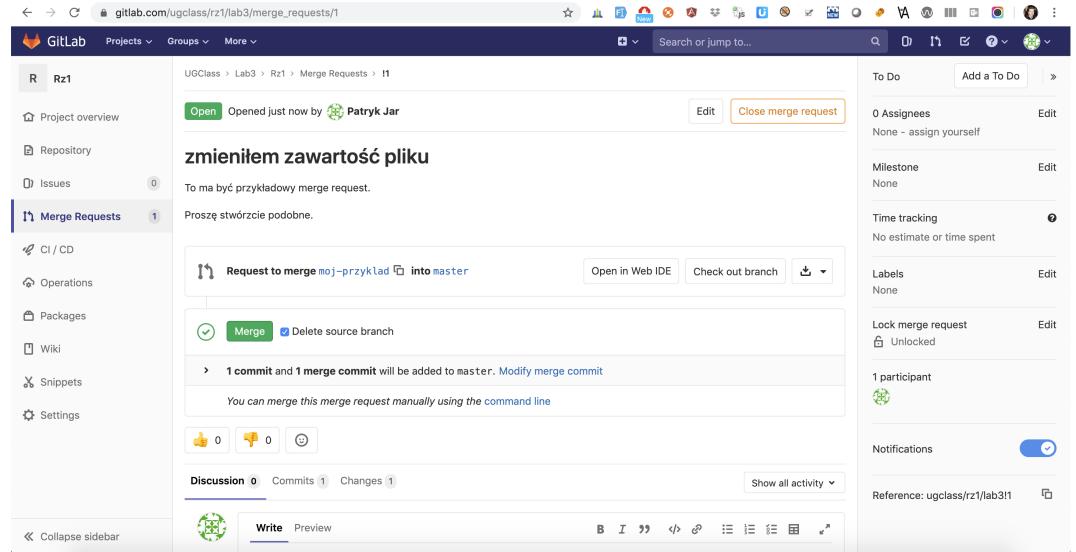


### Tworzenie merge requesta





### Merge request





# Stashowanie plików polecenie: git stash

Polecenie zapisuje śledzone pliki w schowku, z którego możemy je później odzyskać.

git stash – zapisuje wszystkie śledzone pliki do schowka

git stash list – wyświetla listę zmian ze schowka

git stash apply <i> – aplikuje zmiany ze schowka

git stash pop <i> – aplikuje zmiany ze schowka i usuwa je z listy zmian

git stash drop <i> – usuwa zmiany ze schowka

