



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 <branch> – utworzenie i przełączenie się na nowego brancha  
$ cd <file_name> – przejście do katalogu  
$ ls – 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 <branch> – 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

Konflikty

polecenie: **git push**

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.

```
[Satyr:ug_test_p1 karolp$ git push
To https://gitlab.com/karolpisarek-kainos/ug_test_p1.git
! [rejected]        master -> master (non-fast-forward)
error: failed to push some refs to 'https://gitlab.com/karolpisarek-kainos/ug_test_p1.git'
hint: Updates were rejected because the tip of your current branch is behind
hint: its remote counterpart. Integrate the remote changes (e.g.
hint: 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
```

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 😊

```
[Satyr:ug_test_p1 karolp$ git blame equation.txt
00000000 (Not Committed Yet 2019-11-19 12:04:28 +0100 1) <<<<<< HEAD
2592385b (karolpisarek-kainos 2019-11-19 12:00:43 +0100 2) f(y)=a*x-b
00000000 (Not Committed Yet 2019-11-19 12:04:28 +0100 3) =====
9847d29d (Julia Czarnowska 2019-11-19 11:56:03 +0100 4) f(y)=2a*x+b
00000000 (Not Committed Yet 2019-11-19 12:04:28 +0100 5) >>>>>> 9847d29d2d26cfa264b8add28ece054211efaf9d
```

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 <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 jesteś na **masterze**.
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.
6. Zmerge'uj swojego brancha do **mastera** (tak, najpierw 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

```
[americano:lab3 patrykj$ git status
On branch moj-przyklad
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   Gr1/Zad3/zad_3.txt

no changes added to commit (use "git add" and/or "git commit -a")
[americano:lab3 patrykj$ cat Gr1/Zad3/zad_3.txt
Mam na imię Patryk
Dzisiaj jest ładny dzień
Moim ulubionym kolorem jest #fff
americano:lab3 patrykj$
```

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

- R Rz1
- Project overview
 - Repository
 - Issues 0
 - Merge Requests 0
 - CI / CD
 - Operations
 - Packages
 - Wiki
 - Snippets
 - Settings
- ⏪ Collapse sidebar

UGClass > Lab3 > Rz1 > Merge Requests > New

New Merge Request

From **moj-przyklad** into **master** [Change branches](#)

Title

Start the title with **WIP:** to prevent a **Work In Progress** merge request from being merged before it's ready.
Add [description templates](#) to help your contributors communicate effectively!

Description

Write Preview

B *I* ” </> 🔗 ☰ ☷ ☸ ☹ ☺ ☻ ☼ ☽ ☾ ☿ ♀ ♂ ♋ ♌ ♍ ♎ ♏ ♐ ♑ ♒ ♓

Describe the goal of the changes and what reviewers should be aware of.

Markdown and quick actions are supported [Attach a file](#)

Assignee

Unassigned

Assign to me

Milestone

Milestone

Merge request

← → ↻ gitlab.com/ugclass/rz1/lab3/merge_requests/1 ☆

GitLab Projects Groups More + Search or jump to... 🔍 🏠 ↺ 📄 ? 🌐

R Rz1

🏠 Project overview

📄 Repository

📄 Issues 0

🔗 Merge Requests 1

🔗 CI / CD

🔗 Operations


📦 Packages

📖 Wiki

✂ Snippets

⚙ Settings

UGClass > Lab3 > Rz1 > Merge Requests > !1

Open Opened just now by  Patryk Jar

Edit

Close merge request

zmieniłem zawartość pliku

To ma być przykładowy merge request.

Proszę stwórzcie podobne.

🔗 Request to merge `moj-przyklad` into `master`

Open in Web IDE

Check out branch



Merge

☒ Delete source branch

> 1 commit and 1 merge commit will be added to master. [Modify merge commit](#)

You can merge this merge request manually using the [command line](#)



0



0



Discussion 0

Commits 1

Changes 1

Show all activity ▾



Write

Preview

B *I* “ ” </> 🔗 ⋮ ⋮ ⋮ ⋮ ↗

To Do

Add a To Do



0 Assignees

Edit

None - assign yourself

Milestone

Edit

None

Time tracking



No estimate or time spent

Labels

Edit

None

Lock merge request

Edit

🔒 Unlocked

1 participant



Notifications



Reference: ugclass/rz1/lab3!1



⏮ Collapse sidebar

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