Sprawozdanie z laboratorium #6

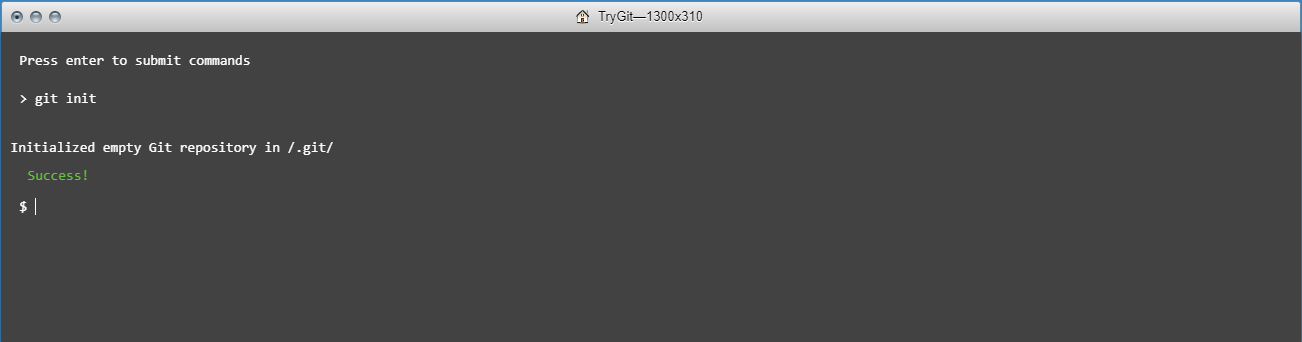
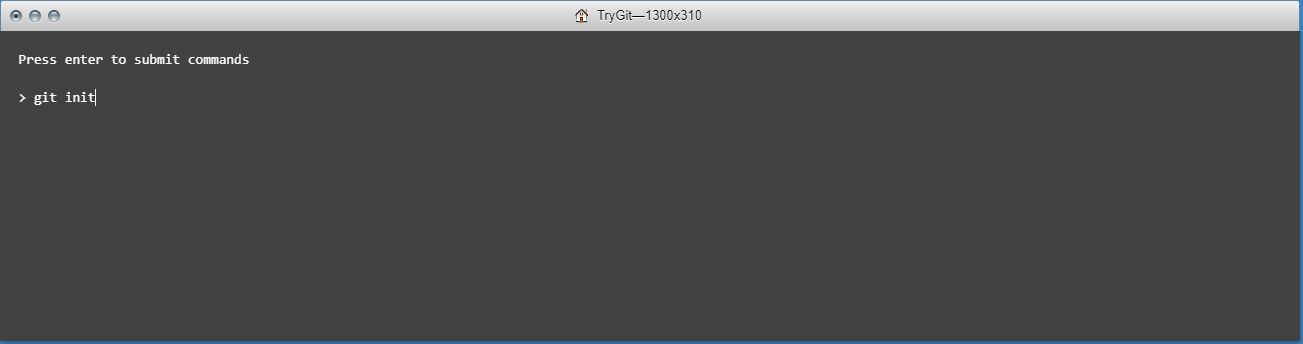
GitHub

Alan Stocki

# Przejście przez kurs na stronie <https://try.github.io>

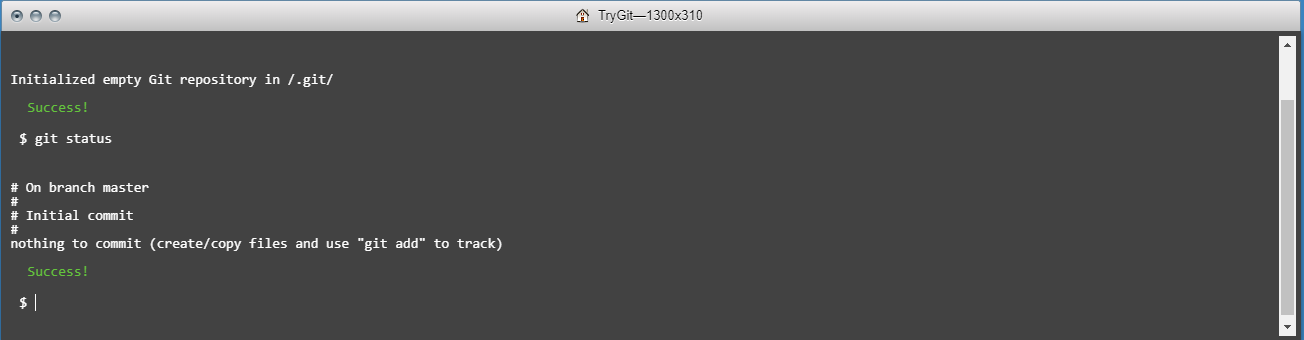
## Krok 1

Inicjalizacja repozytorium Git



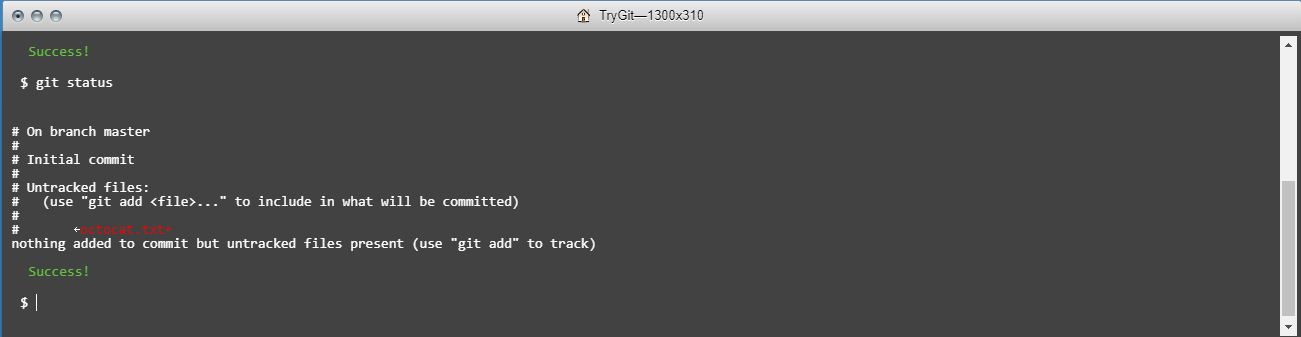
## Krok 2

Sprawdzenie statusu repozytorium



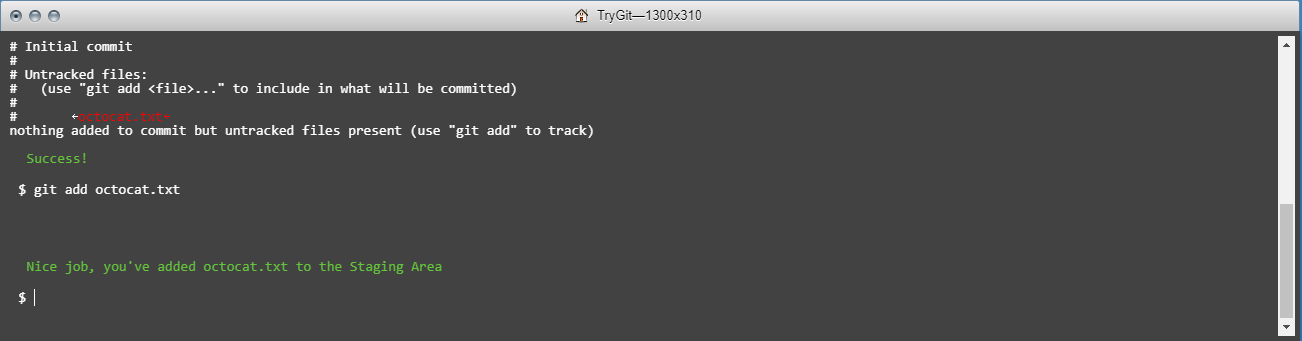
## Krok 3

Sprawdzenie statusu repozytorium po dodaniu pliku octocat.txt



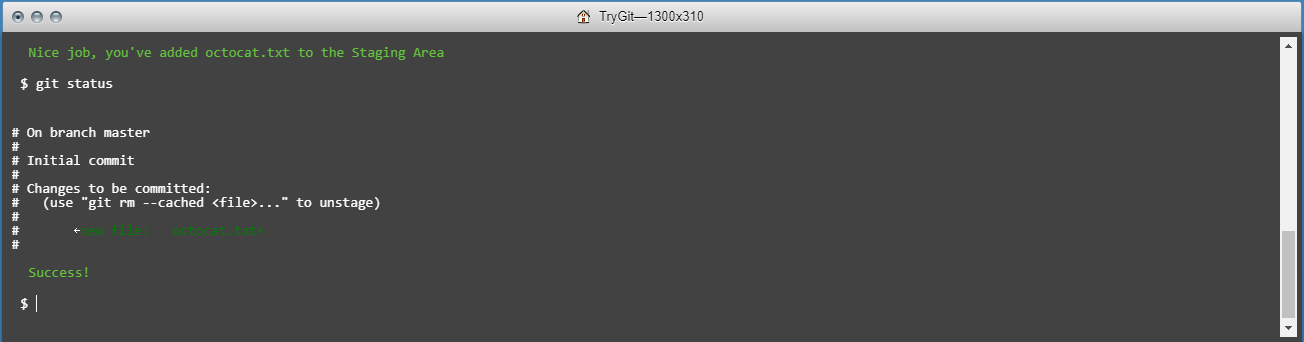
Trzeba dodać w kolejnym kroku plik octocat.txt do śledzenia za pomocą komendy „git add octocat.txt”

## Krok 4



## Krok 5

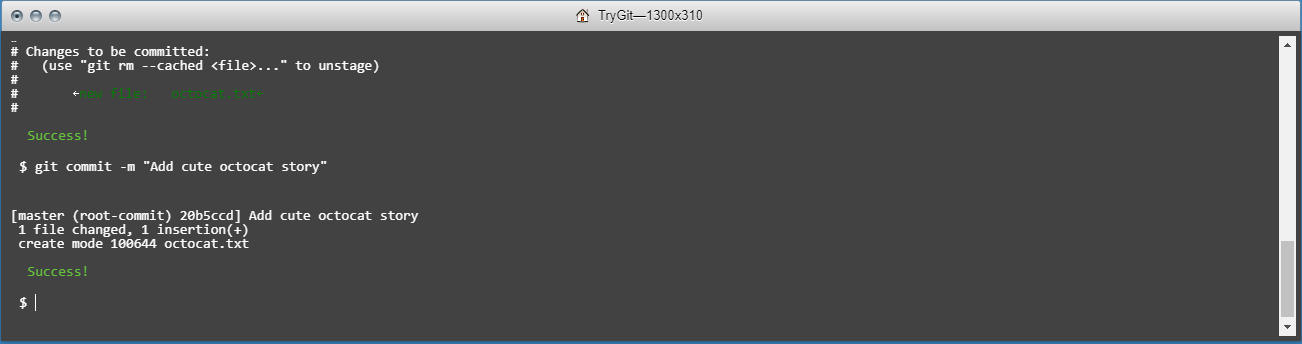
Ponowne sprawdzenie statusu



Udało się, plik octocat.txt został pomyślnie dodany do Staging area i zmiany w tym pliku od teraz będą śledzone. (Nie znajduje się jeszcze plik w repozytorium!)

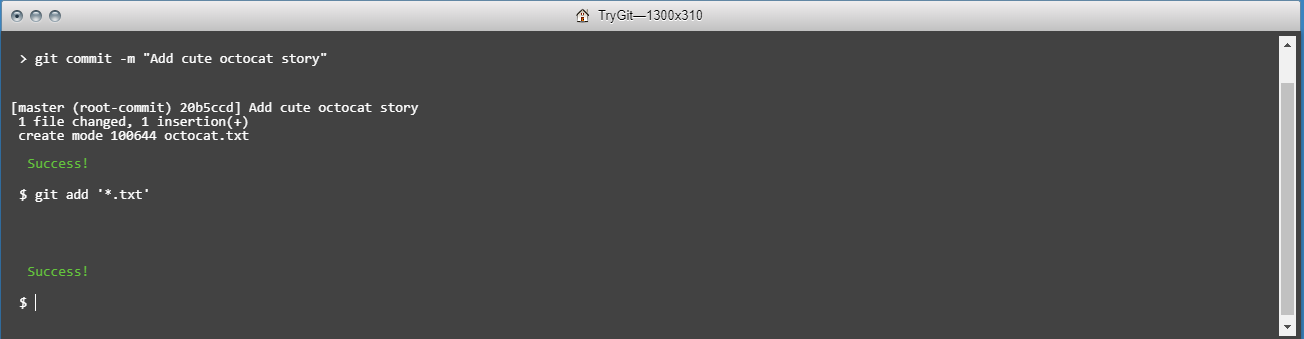
## Krok 6

Robimy Commit pliku za pomocą komendy: > git commit -m "Add cute octocat story"



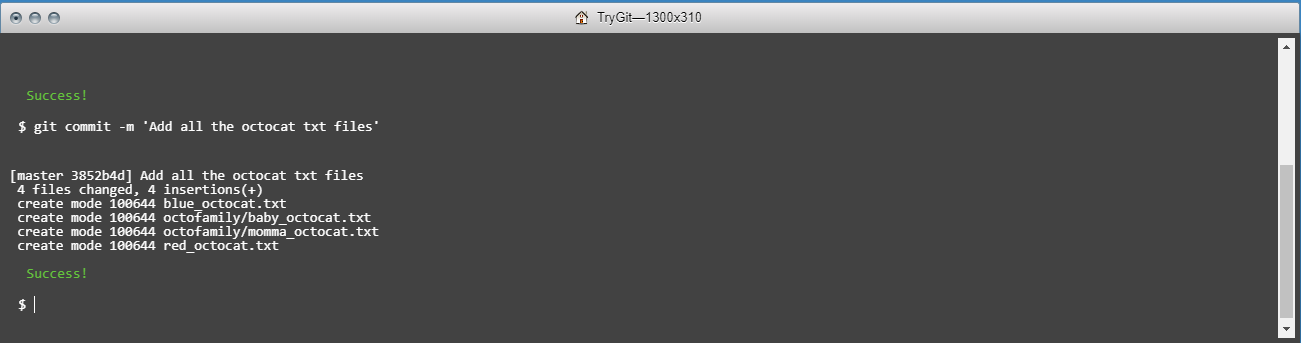
## Krok 7

Użycie „Dzikiej karty” (\*) aby dodać wszystkie pliki o tym samym rozszerzeniu (‘\*.txt’)



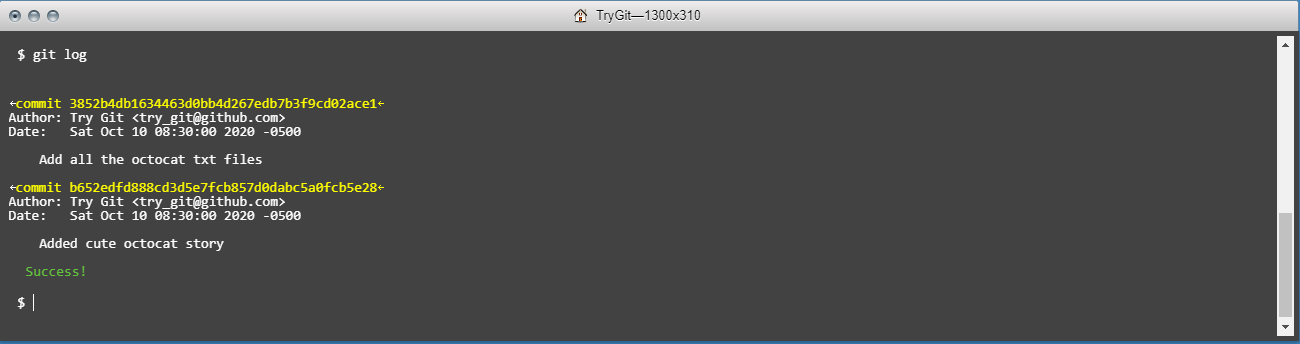
## Krok 8

Commitowanie wszystkich dodanych plików



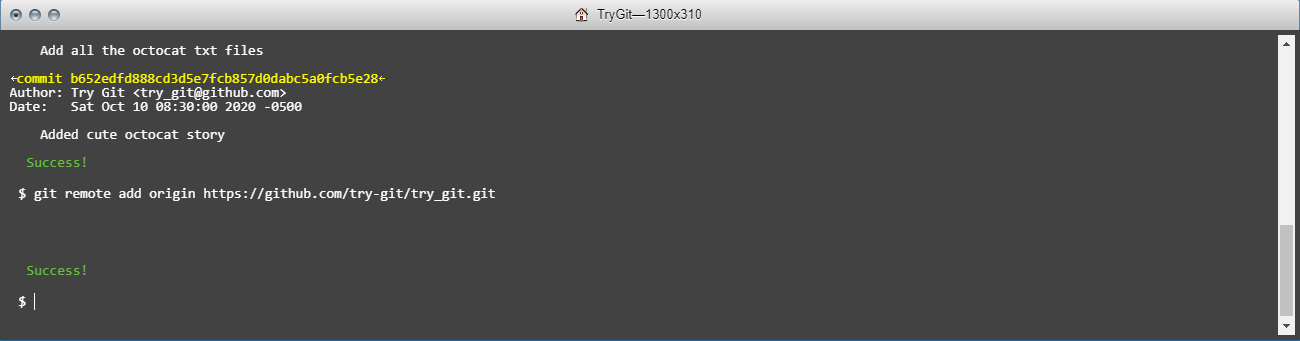
## Krok 9

Użycie git log’u aby zobaczyć listę wszystkich zmian



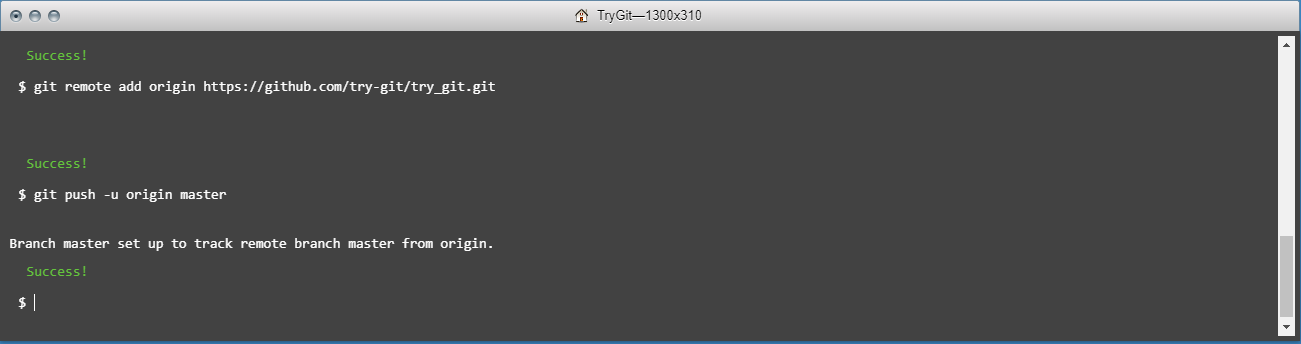
## Krok 10

Tworzenie zdalnego repozytorium na stronie <https://github.com/try-git/try_git.git>



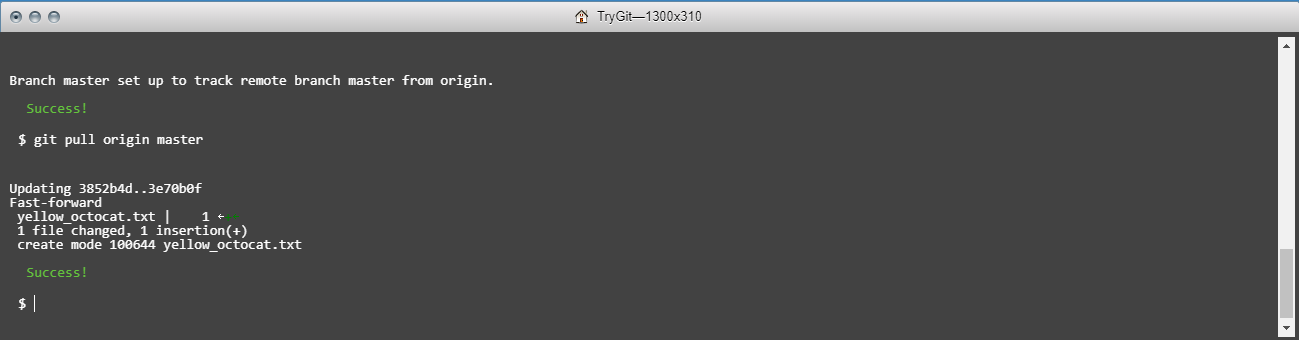
## Krok 11

Zdalne pushowanie naszych commitów do origin repo na GitHubie ;)



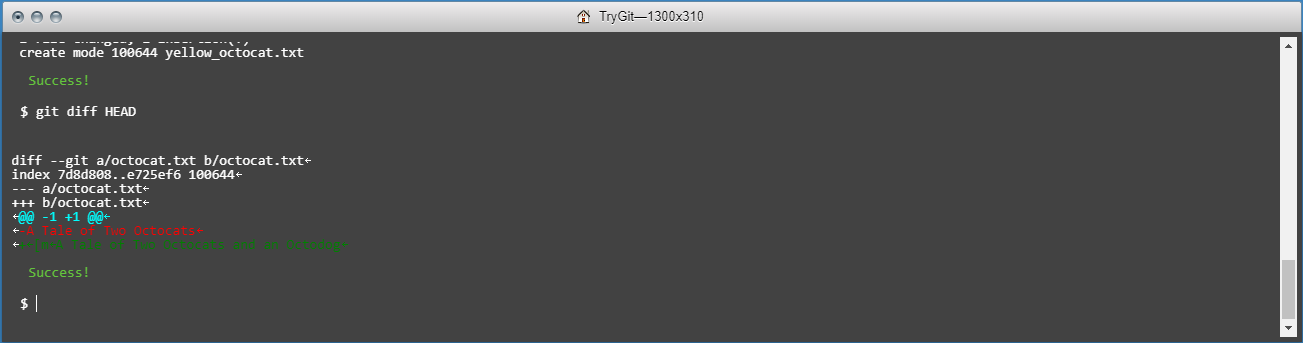
## Krok 12

Pull’owanie zmian zaszłych w naszym repo



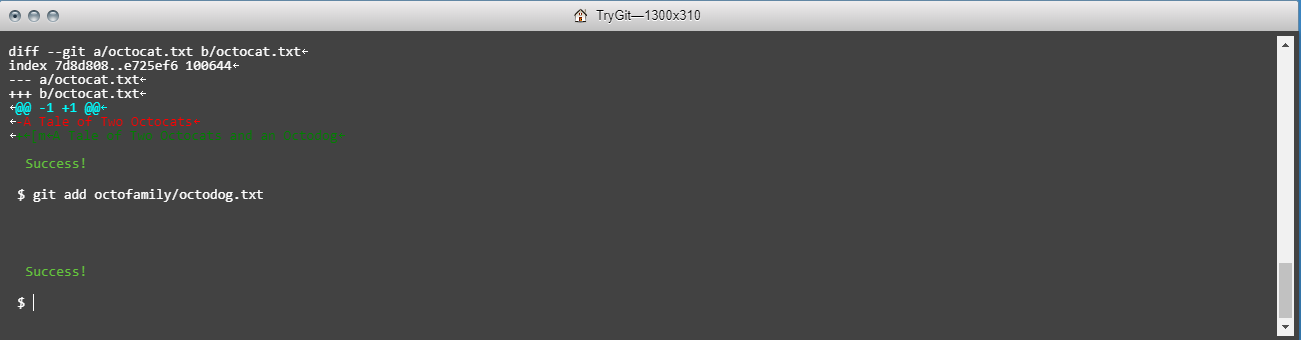
## Krok 13

Sprawdzenie różnic w plikach po pull’u przy pomocy > git diff HEAD (HEAD – pozwala na sprawdzenie zmian w najnowszym commicie)



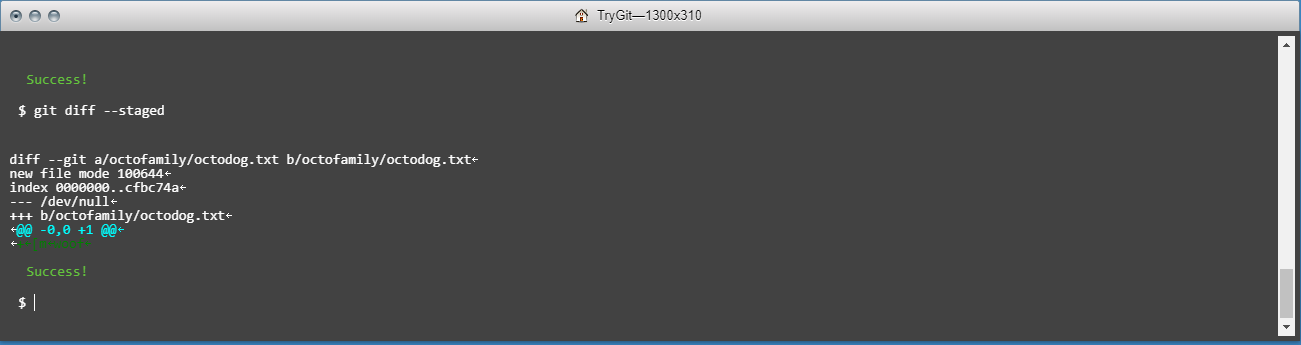
## Krok 14

Sprawdzanie zmian w plikach stage’owanych  
Najpierw dodajemy plik „octodog.txt”



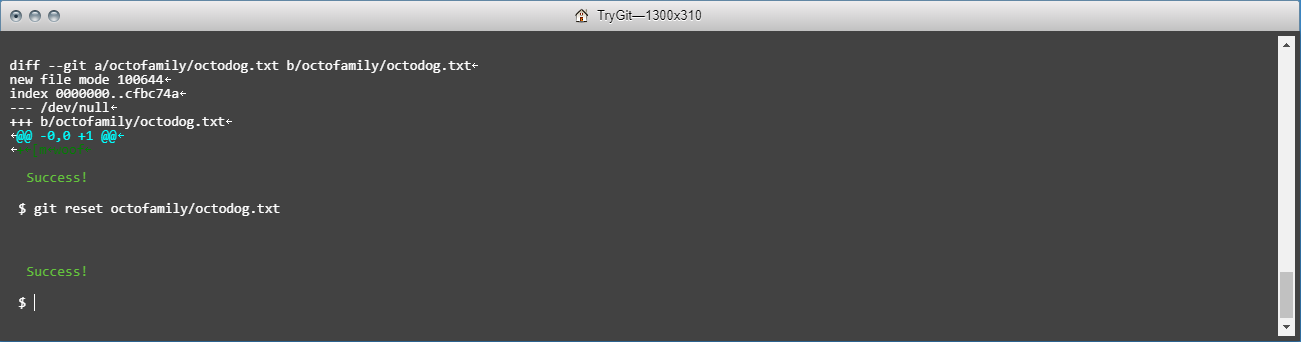
## Krok 15

Sprawdzenie zmian w stage’owanych plikach > git diff –staged (--staged – opcja wyświetlania zmian w plikach stage’owanych)



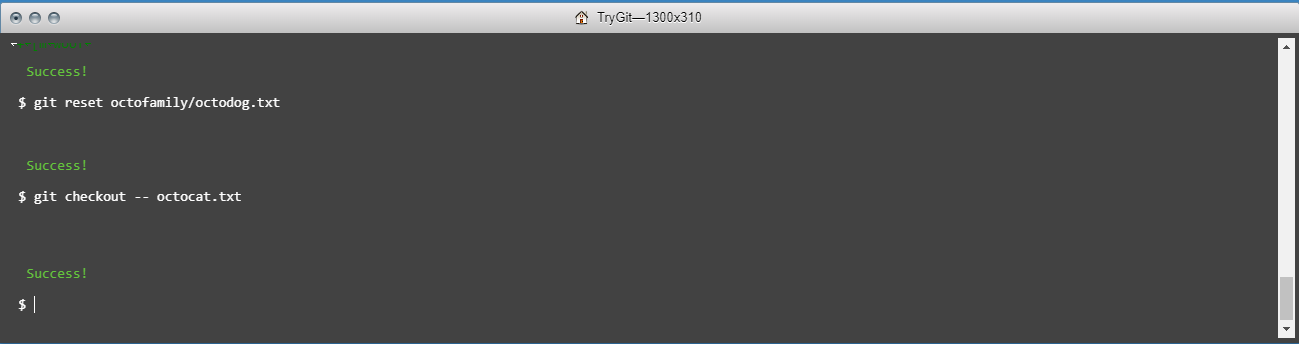
## Krok 16

Usuwanie pliku octodog.txt z plików stage’owanych przy pomocy komendy:  
$ git reset octofamily/octodog.txt



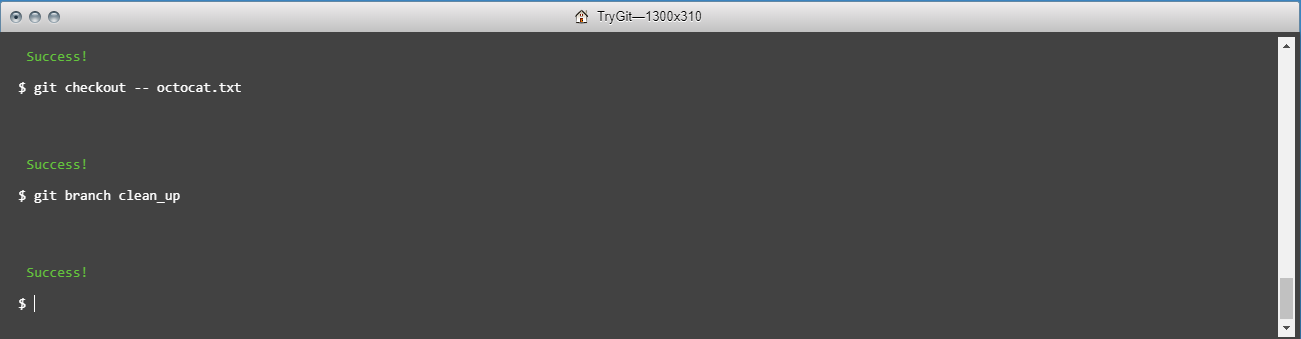
## Krok 17

Przywrócenie stanu do momentu ostatniego commit’a pliku „octocat.txt”



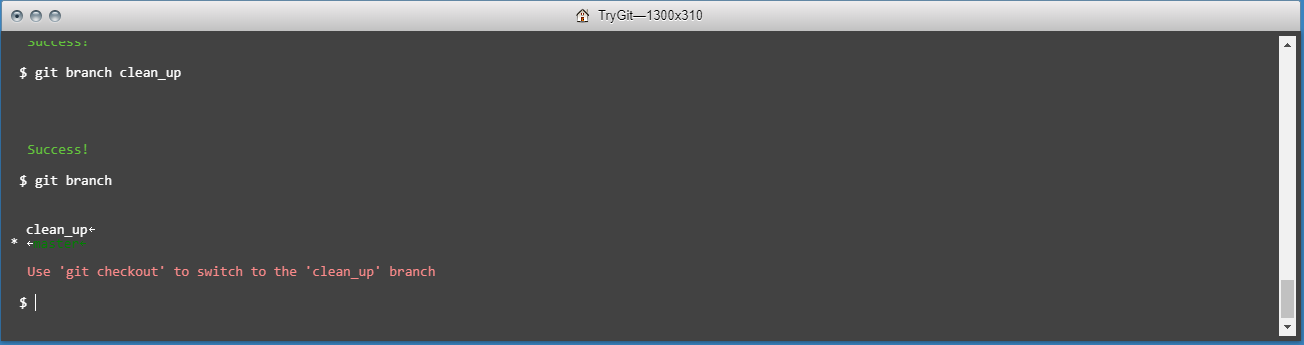
## Krok 18

Tworzenie gałęzi w celu „posprzątania”



## Krok 18.1

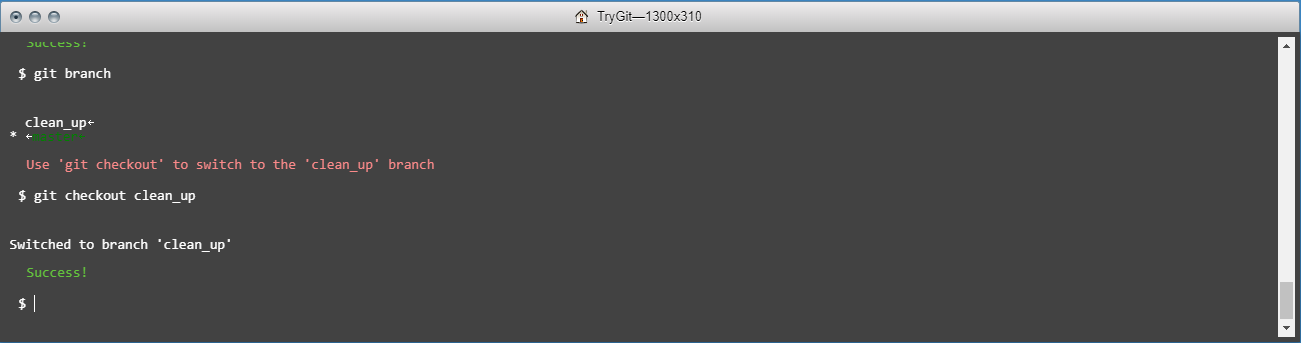
Sprawdzenie gałęzi



Faktycznie, istnie master oraz clean\_up.

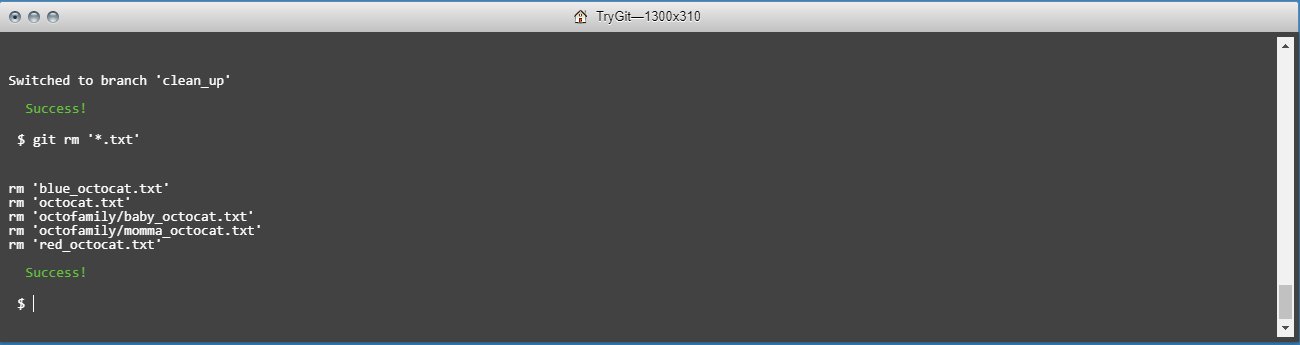
## Krok 19

Switch do gałęzi „clean\_up”



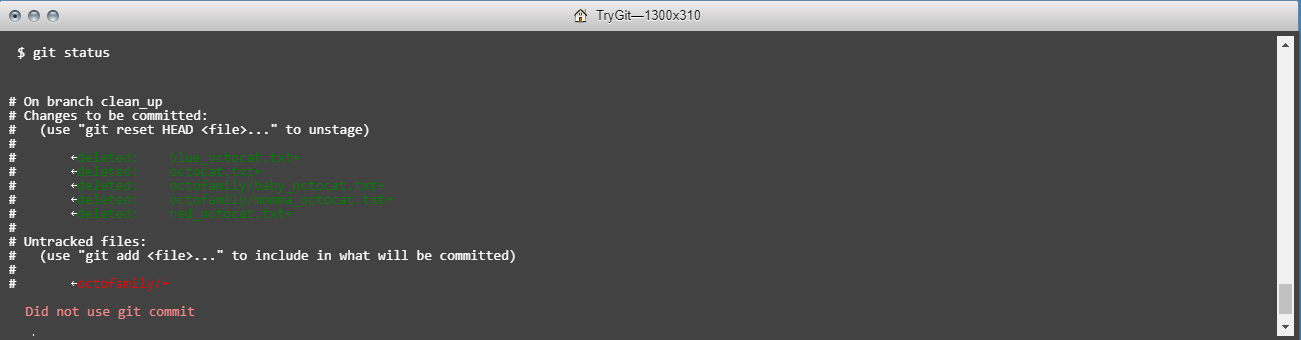
## Krok 20

Usunięcie plików z dysku oraz stage’u przy pomocy komendy > git rm '\*.txt'



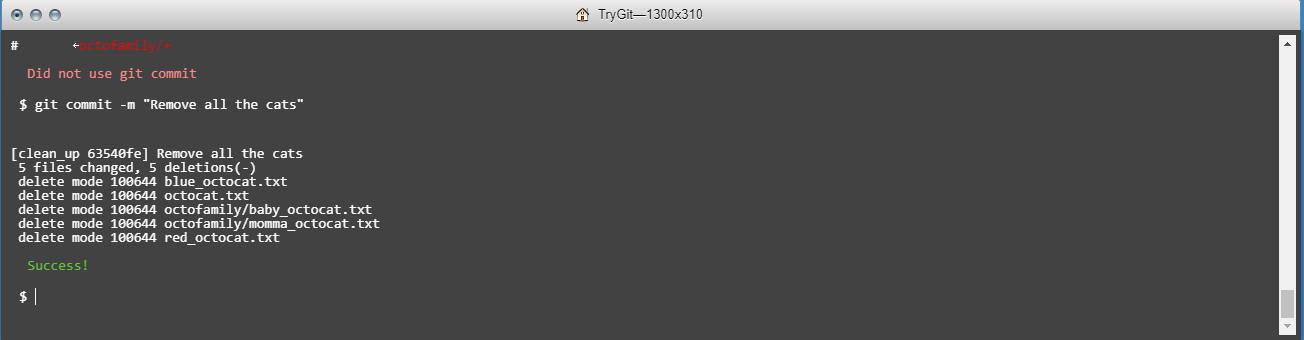
## Krok 21.1

Sprawdzenie statusu przed commitem



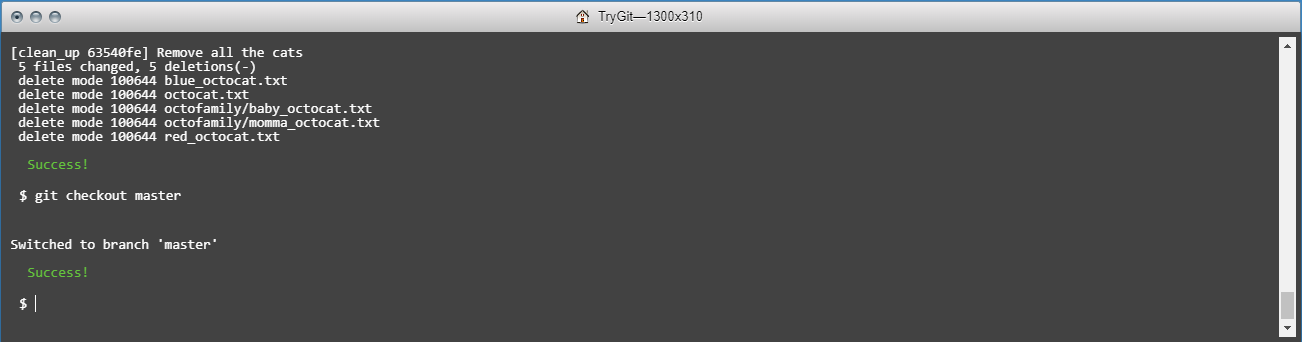
## Krok 21.2

Commit zmian



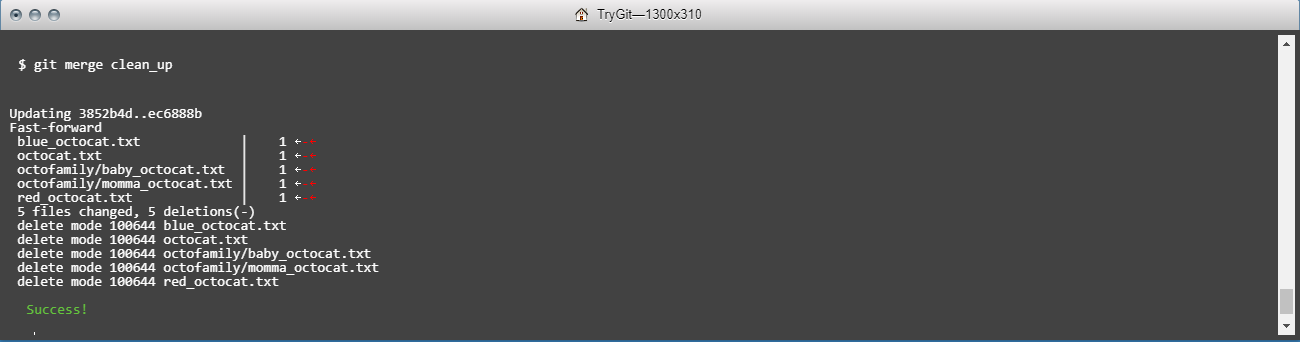
## Krok 22

Powrót do gałęzi ‘master’



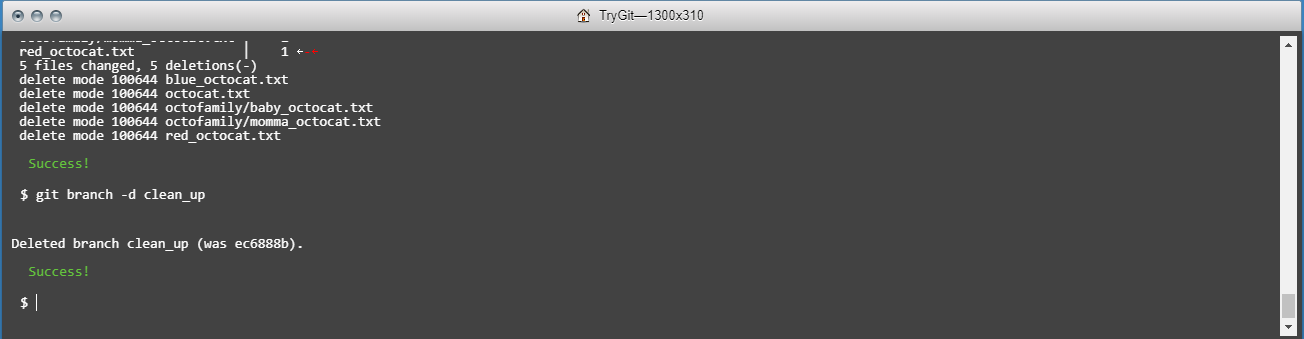
## Krok 23

Merge’owanie gałęzi clean\_up z ‘master’



## Krok 24

Usunięcie niepotrzebnej gałęzi, w tym wypadku: clean\_up



## Krok 25

Finalny push zmian

