

GIT & BASH

KOŁO NAUKOWE DATA SCIENCE

Natalia Potocka

26.04.2017

INSTALACJA

- Aby zainstalować program Git Bash, wejdź na tę stronę <http://git-scm.com/downloads> i ściągnij plik dla Twojego systemu operacyjnego.
- Aby założyć konto na GitHubie, wejdź na stronę <https://github.com> i zarejestruj się. Najlepiej podaj email wydziałowy (studencki), ponieważ umożliwi Ci on wystąpienie o konto edukacyjne. Jeśli jednak podałeś inny email, zawsze możesz dodać kolejny w ustawieniach swojego konta.
- Aby wystąpić o konto edukacyjne, wejdź na stronę <http://education.github.com>, kliknij Request a discount, wypełnij potrzebne pola (koniecznie podaj studencki email!), a za parę dni powinieneś dostać odpowiedź o pozytywnym rozpatrzeniu wniosku. Dzięki temu będziesz mógł założyć do 5 prywatnych repozytoriów, tj. takich, których nie widzi nikt poza Tobą i wybranymi przez Ciebie osobami.

BASH - KONSOLA LINUXOWA

Bash is a command interpreter that allows the execution of commands as well as redirecting their in- and outputs and a programming language, which provides built-in commands, control structures, shell functions, shell expansion, shell redirection or recursive programming.

BASH

Gdy uporasz się z instalacją, otwórz program Git Bash.

Każde polecenie zatwierdza się ENTEREM. Po zakończeniu wykonywania polecenia pojawia się \$, oznaczający, że użytkownik może podać kolejne polecenie.

Przydatne skróty:

- CTRL+C - przerywa wykonywanie polecenia
- →, ←, Home, End - nawigacja w wierszu
- ↑, ↓ - przeglądanie historii komend
- tab - uzupełnia wpisywane polecenie

PODSTAWOWE KOMENDY - FOLDERY

\$ `ls` # lista plików - komenda `ls`

\$ `ls -a` # lista wszystkich plików
komenda `ls` z argumentem opcjonalnym `-a`

\$ `pwd` # print working directory
wypisz folder, w którym się znajdujesz

\$ `cd katalog` # change directory

\$ `..` # parent directory - wejdź folder wyżej

\$ `mkdir katalog` # make directory - utwórz nowy folder

\$ `rmdir katalog` # remove directory - usuń folder

\$ `rm -r katalog` # j.w.

PODSTAWOWE KOMENDY - PLIKI

\$ `cp` plik1 plik2 # copy - kopiowanie pliku

\$ `mv` co dokad # przenieś plik "co" do folderu "dokad"

\$ `rm` plik # usuwanie pliku

\$ `rm /` # czyści wszystkie pliki na dysku !!!!

\$ `echo "tekst" > plik` # przekierowanie do nowego pliku
(plik będzie miał treść "tekst")

\$ `touch` plik # stworzenie pustego pliku o nazwie plik

System kontroli wersji śledzi wszystkie zmiany dokonywane na pliku (lub plikach) i umożliwia przywołanie dowolnej wcześniejszej wersji. (...) Pozwala on przywrócić plik(i) do wcześniejszej wersji, odtworzyć stan całego projektu, porównać wprowadzone zmiany, dowiedzieć się kto jako ostatnio zmodyfikował część projektu powodującą problemy, kto i kiedy wprowadził daną modyfikację. Oprócz tego używanie VCS oznacza, że nawet jeśli popełnisz błąd lub stracisz część danych, naprawa i odzyskanie ich powinno być łatwe. Co więcej, wszystko to można uzyskać całkiem niewielkim kosztem.

ZAKŁADANIE REPOZYTORIUM

Założ repozytorium (zrobi się w tym folderze, w którym aktualnie pracujesz):

```
$ git init
```

Musisz skonfigurować repozytorium, aby Git „wiedział”, że to Ty nad nim pracujesz:

```
$ git config --global user.name "UserName"  
    # przypisz swoją nazwę użytkownika  
    # do repozytorium  
$ git config --global user.email "User@email.com"  
    # przypisz email  
    # (ten, który podałeś na GitHubie)
```

Napis (master) na końcu wiersza z obecnym katalogiem oznacza, że znajdujesz się w repozytorium.

STATUS PLIKÓW

Sprawdź aktualny status plików:

- untracked (nieobserwowane),
- unstaged (obserwowane, zmienione),
- staged (obserwowane i zmiany zapisane),
- deleted (usunięte)

```
$ git status
```

DODAWANIE PLIKÓW

Dodaj plik, tak, żeby system obserwował jego zmiany (wtedy status zmienia się na staged):

```
$ git add plik  
$ git add --all # (lub git add -A)  
                # dodaj wszystkie pliki
```

Gdy usunąłeś plik musisz także „poinformować” Gita, że go usunąłeś:

```
$ git rm plik
```

COMMITOWANIE

Następnie należy zrobić „commita”, tzn. przygotować pliki do oddania je na serwer (*ang. commit* - dopełnić, oddać, powierzyć):

```
$ git commit -m "krótki opis zmian/wykonanej pracy"
```

Ważne: wiadomość zawsze musi się pojawić. Zazwyczaj jest to krótki opis wykonanej pracy lub dokonanych zmian w plikach.

REPOZYTORIUM ZDALNE (NA SERWERZE)

Do tej pory repozytorium znajduje się tylko na komputerze. Aby połączyć je z kontem na GitHubie, należy stworzyć repozytorium na swoim koncie.

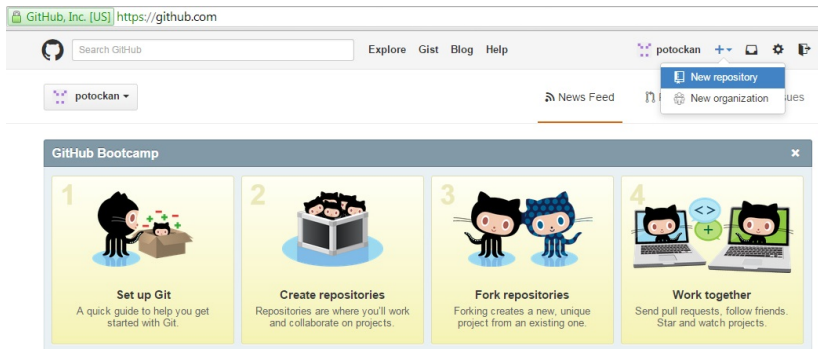


FIGURE 1: NewRepo

ŁĄCZENIE REPO ZDALNEGO Z LOKALNYM

Po przejściu przez proces zakładania zdalnego repozytorium (należy podać nazwę i wybrać czy jest ono publiczne czy prywatne, w tym przypadku opcję Initialize this repository with a README pozostawiamy niezaznaczoną), trzeba połączyć nasze lokalne repozytorium z tym zdalnym:

```
$ git remote add origin  
https://github.com/potockan/Repo.git
```

Oczywiście, adres dla każdego użytkownika i dla każdego nowego repozytorium będzie inny. W następnym kroku powinien się wyświetlić adres, który należy podać.

Quick setup — if you've done this kind of thing before



Set up in Desktop

or

HTTP

SSH

https://github.com/potockan/Repo.git

We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

ŁĄCZENIE REPO ZDALNEGO Z LOKALNYM

Innym sposobem założenia repozytorium, jest najpierw założenie go na GitHubie, a dopiero potem na swoim komputerze. Proces wygląda podobnie, z tą tylko różnicą, że tym razem zaznacz opcję `Initialize this repository with a README`. Wówczas zostaniesz od razu przeniesiony na stronę gotowego repozytorium. Aby połączyć je ze swoim komputerem, należy w konsoli (Git Bash) wkleić komendę:

```
$ git clone https://github.com/potockan/repo3.git
```

ŁĄCZENIE REPO ZDALNEGO Z LOKALNYM

Dokładny adres do wklejenia podany jest na stronie repozytorium.

The screenshot shows the GitHub interface for a repository named 'repo3' by user 'potockan'. At the top, there are navigation links for 'potockan / repo3' and buttons for 'Unwatch', 'Star', and 'Fork'. Below this is a 'Description' section with input fields for a short description and a website, along with 'Save' and 'Cancel' buttons. A summary bar shows '1 commit', '1 branch', '0 releases', and '1 contributor'. The main content area displays the 'Initial commit' by 'potockan' a minute ago, with a commit hash 'e4f6fd47e1'. Below the commit is a file named 'README.md'. On the right sidebar, there are links for 'Code', 'Issues', 'Pull Requests', 'Wiki', 'Pulse', 'Graphs', and 'Settings'. The 'Code' section is expanded, showing the 'HTTPS clone URL' as 'https://github.com/potockan/repo3.git', which is highlighted with a red box. Below the URL, it says 'You can clone with HTTPS, SSH, or Subversion'. At the bottom of the sidebar are buttons for 'Clone in Desktop' and 'Download ZIP'.

potockan / repo3

Unwatch 1 Star 0 Fork 0

Description Website

Short description of this repository Website for this repository (optional) Save or Cancel

1 commit 1 branch 0 releases 1 contributor

branch: master repo3 / +

Initial commit

potockan authored a minute ago latest commit e4f6fd47e1

README.md Initial commit a minute ago

README.md

repo3

Code

Issues 0

Pull Requests 0

Wiki

Pulse

Graphs

Settings

HTTPS clone URL

https://github.com/potockan/repo3.git

You can clone with HTTPS, SSH, or Subversion

Clone in Desktop

Download ZIP

WYSYŁANIE NA SERWER

Wracając do commitów. Masz już przygotowane pliki, żeby wysłać je na serwer, tzn. zrobić „commita”. Teraz trzeba je tylko wysłać. Służy do tego komenda:

```
$ git push # lub  
$ git push origin master  
# origin master jest opcjonalne
```

Komenda ta wysyła na GitHuba wszystkie dodane zmiany. Jeśli ktoś z Tobą pracuje w tym samym repozytorium, przed pushem należy zrobić pulla:

```
$ git pull # lub  
$ git pull origin master  
# origin master jest opcjonalne
```

Komenda ta ściąga z GitHuba wszystkie zmiany i zapisuje na Twoim dysku.

DODAWANIE WSPÓŁPRACOWNIKÓW

Aby dodać więcej osób do Twojego repozytorium, trzeba zmienić to w ustawieniach repozytorium.

The screenshot displays the GitHub interface for a repository named 'repo3' by user 'potockan'. The main content area shows the 'Initial commit' with a 'README.md' file. A large red arrow points from the repository name 'repo3' towards the bottom of the page. On the right sidebar, the 'Settings' option is highlighted with a red rectangle. At the bottom of the page, the 'Collaborators' section is also highlighted with a red rectangle, showing a text input field with the text 'norbertryciak' and an 'Add collaborator' button.

branch: master repo3 / +

Initial commit

potockan authored 4 hours ago latest commit e4f6fd47e1

README.md Initial commit 4 hours ago

repo3

Settings

HTTPS clone URL
https://github.com/potockan/repo3.git

You can clone with HTTPS, SSH, or Subversion

This repository Search Explore Gist Blog Help

potockan + Fork 0

Options

Collaborators

Full access to the repository

This repository currently has no collaborators.

norbertryciak Add collaborator

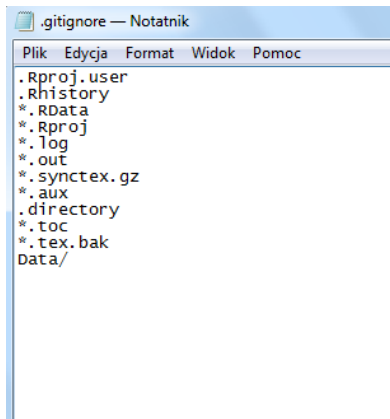
.GITIGNORE

Warto jeszcze wiedzieć o pliku `.gitignore`. Służy on do wylistowania plików, których nie chcemy wysyłać na GitHuba, gdyż są to na przykład duże dane lub tzw. „śmieci” (np. przy kompilacji plików TeXowych pojawia się dużo dodatkowych plików, które nie są potrzebne).

.GITIGNORE

Przykład:

W tym przypadku wszystkie pliki o rozszerzeniu `.RData`, `.Rproj`, `.log`, `.out`, `.synctex.gz`, `.aux`, `.toc`, `.tex.bak` (te z gwiazdkami przed kropką) z każdego katalogu będą ignorowane przez Git. Co więcej, wszystkie pliki z katalogu `Data` również nie będą obserwowane. Również pliki o rozszerzeniu `.Rproj.user`, `.Rhistory`, `.directory` będą ignorowane, ale tylko w katalogu głównym repozytorium.



```
.Rproj.user
.Rhistory
*.RData
*.Rproj
*.log
*.out
*.synctex.gz
*.aux
.directory
*.toc
*.tex.bak
Data/
```

POWRÓT DO POPRZEDNIEGO COMMITA

Aby powrócić do poprzedniego commita, trzeba znać ich identyfikatory. Aby je wyświetlić mamy komendę:

```
npotocka@gwp:~/priv/nau...taty bash/super_repo$ git log master
commit ff647b94d4ef2d124666af70ccb22367c3fc7cd3
Author: potockan <potockan@gmail.com>
Date:   Wed Apr 26 19:23:40 2017 +0200

    dodajemy .gitignore

commit f6d10a925539d080b969617ff62997a05b7394e8
Author: potockan <potockan@gmail.com>
Date:   Wed Apr 26 19:13:11 2017 +0200

    Revert "dodaje plik3"

    This reverts commit 7ec91a46552e4a08d4b4238afcac43581c9cfab5.

    powrot do pliku1 i pliku2

commit f41cc258fc35811c4b893190a71084fad6ea1047
Author: potockan <potockan@gmail.com>
Date:   Wed Apr 26 19:09:03 2017 +0200

    dodaje plik4

commit 7ec91a46552e4a08d4b4238afcac43581c9cfab5
Author: potockan <potockan@gmail.com>
Date:   Wed Apr 26 19:08:00 2017 +0200

    dodaje plik3

commit 74c1a48b53fb3b2a0d5f70f544952643d742ac3d
Author: potockan <potockan@gmail.com>
Date:   Wed Apr 26 19:07:25 2017 +0200

    dodaje plik2
```

\$ git log master

POWRÓT DO POPRZEDNIEGO COMMITA

A następnie możemy przywrócić commit, który nas interesuje

```
$ git revert 7ec91a46552e4a08d4b4238afcac43581c9cfab5
```

```
$ git add --all
```

```
$ git push origin master
```

Przywrócenie poprzedniego commita **nie powoduje usunięcia go** z historii, lecz dodanie kolejnego commita z cofniętymi zmianami. Usunięciu danego commita z historii służy komenda `git revert`, jednak jest ona nie polecana.

Tworzenie gałęzi

Aby stworzyć nową gałąź mamy komendę:

```
$ git checkout -b NAZWA_GALEZI
```

Po dokonaniu zmian, aby wrzucić je na serwer wykonujemy:

```
$ git add --all # lub część plików
```

```
$ git commit -m "dodajemy pliki"
```

```
$ git push origin NAZWA_GALEZI # nie master!
```

Aby przełączać się między gałęziami używamy polecenia:

```
$ git checkout NAZWA_GALEZI # może być git checkout master
```

ŁĄCZENIE GAŁĘZI

Gałęzie można połączyć lokalnie za pomocą polecenia:

```
$ git merge NAZWA_GALEZI/master
```

lub na serwerze używając polecenia Create pull request na githubie.

POWODZENIA :)