|  |
| --- |
| GIT(HUB) – NAUKA |
| Podstawowe komendy |
|  |
| [Wpisz tutaj streszczenie dokumentu. Streszczenie to zwykle krótkie podsumowanie zawartości dokumentu. Wpisz tutaj streszczenie dokumentu. Streszczenie to zwykle krótkie podsumowanie zawartości dokumentu.] |
|  |

GIT(HUB) – NAUKA

Podstawowe komendy

# GITHUB

• • •

Jest to hostingowy serwis internetowy przeznaczony dla projektów programistycznych wykorzystujących system kontroli wersji Git. Github udostępnia także usługę gist, służącą do szybkiego tworzenia małych repozytoriów (są to pełnoprawne repozytoria gita).

Wszystko o plikach i katalogach.

Do tworzenia katalogów wykorzystywane jest polecenie mkdir. Do wykonania polecenia wymagane są uprawnienia administratora.

Dominik@weirdFISH\_ MINGW64 ~

$ mkdir nowy\_folder

Należy zaznaczyć iż wpisanie *$ mkdir* ***nowy folder***spowoduje stworzenie dwóch folderów. Pierwszego ***nowy*** drugiego***folder***.

Nowy plik tworzymy wpisując polecenie:

Dominik@weirdFISH\_ MINGW64 ~

$ touch plik.txt

Jest to bezpieczna metoda, gdyż w przypadku, gdy plik o takiej nazwie już istnieje nie zostaje on nadpisywany przez nowo powstały. Inną opcją jest wpisanie polecenia

Dominik@weirdFISH\_ MINGW64 ~

$ > plik.txt

Tworzy ono nowy plik.txt nadpisując dane. Jeżeli chcemy dopisać ciąg znaków do istniejącego już pliku nie używając edytora możemy to zrobić tak:

Dominik@weirdFISH\_ MINGW64 ~

$ echo 123 >> plik.txt

W ten sposób do istniejącego już w pliku ciągu znaku zostaje dopisany nowy w nowej linijce.

Do edycji plików tekstowych wykorzystywany jest edytor Vim. Aby zacząć edycję pliku wystarczy wpisać.

Dominik@weirdFISH\_ MINGW64 ~

$ vim nazwa\_pliku.txt

W przypadku gdy podany plik nie istnieje zostanie on automatycznie utworzony. Po zatwierdzeniu zostaje włączony edytor:

~

~

~

~

~

~

~/nazwa\_pliku.txt [unix] (00:59 01/01/1970) 0,0-1 All

"nazwa\_pliku.txt" [New File]

Aby zacząć pisać należy nacisnąć **I[insert]** na klawiaturze. Gdy uznamy, że zakończyliśmy pracę wciskamy przycisk **Esc**. Każdą akcję należy poprzedzić znakiem **:**. Tak więc jeżeli chcemy zapisać nasz plik wpisujemy **:w**, jeżeli zapisać i wyjść **:x**. Vim umożliwia również na mapowanie poleceń dzięki czemu możemy ustalić, że np. podwójne kliknięcie przycisku **Esc** spowoduje zapisanie i wyjście z edytora.

# Trudne słówka

• • •

Repozytorium – miejsce uporządkowanego przechowywania dokumentów, z których wszystkie przeznaczone są do udostępniania.

~/nazwa\_pliku.txt [unix] (14:23 08/04/2016) 9,1 All

:map <Esc><Esc> :x<CR>

Takie mapowanie przycisków jest zapisywane tylko w pojedynczej sesji, więc po wyłączeniu edytora ustawienia są resetowane.

Do wyświetlenia ścieżki miejsca w którym się aktualnie znajdujemy wykorzystujemy polecenie **pwd**.

Dominik@weirdFISH\_ MINGW64 ~/nowy\_folder ((adb5885...))

$ pwd

/c/Users/Dominik/nowy\_folder

Aby wyświetlić wszystkie pliki w aktualnej ścieżce wpisujemy polecenie **ls**.

Dominik@weirdFISH\_ MINGW64 ~/nowy\_folder ((adb5885...))

$ ls

123 nowy.txt pliczek2.txt qweqwe README.md

Polecenie **ls** udostępnia również dodatkowe opcje. **–a** wyświetla wszystkie nazwy plików i folderów w aktualnym folderze. **–al** wyświetla wszystkie pliki i foldery wylistowane, wraz z dodatkowymi informacjami. Kolejno są to: **uprawnienia do pliku/folderu(1), liczba łączy(2), ID użytkownika(3), ID grupy(4), liczba bajtów(5), data ostatniej modyfikacji(6), nazwa pliku(7).**

Dominik@weirdFISH\_ MINGW64 ~/nowy\_folder ((adb5885...))

$ ls -l nowy.txt

**(1)**-rw-r--r— **(2)**1 **(3)**Dominik **(4)**197121 **(5)**0 **(6)**kwi 9 18:26 **(7)**nowy.txt

Aby zignorować niektóre pliki wpisujemy **ls –I ”\*.txt, plik\*”**. Tak wprowadzona komenda ukrywa wszystkie pliki z rozszerzeniem .txt oraz pliki i foldery których nazwy zaczynają się słowem „plik”.

Gdy chcemy posortować wynik według terminu modyfikacji wykorzystujemy opcję **–t (-lt)**. Polecenie **ls** umożliwia również wylistowanie wszystkich folderów – **ls -ld \*/** - dla informacji o dostępie lub **ls -Ld \*/** dla wyświetlenia samych nazw folderów.

Dominik@weirdFISH\_ MINGW64 ~

**$ ls -ld \*/**

drwxr-xr-x 1 Dominik 197121 0 mar 4 15:18 AppData//

drwxr-xr-x 1 Dominik 197121 0 kwi 2 14:48 bin//

drwxr-xr-x 1 Dominik 197121 0 lut 11 10:06 Contacts//

drwxr-xr-x 1 Dominik 197121 0 mar 31 14:53 Cookies//

drwxr-xr-x 1 Dominik 197121 0 mar 18 13:24 'Dane aplikacji/'/

Do wylistowania plików bez folderów należy posłużyć się poleceniem:

**find . -maxdepth 1 -type f.**

Usuwanie pliku odbywa się poprzez wpisanie

# GIT początki

Git nie zapewnia uwierzytelniania wprowadzanej treści. Każdy może zmienić pliki – nie ma loginu i hasła do mastera.

Wprowadzane zmiany nie korzystają z czasu globalnego – pliki wprowadzane posiadają lokalny czas modyfikacji przez co ciężko jest określić kto i kiedy dokonał zmian. Możliwe jest jedynie ustalenie która zmiana nastąpiła pierwsza.

Po pierwsze tworzymy nowo uworzony folder jako repozytorium git’a. Aby do tego doszło należy w utworzonym folderze zainicjować git’a:

Dominik@weirdFISH\_ MINGW64 ~/Desktop/jest\_GIT

$ git init

Initialized empty Git repository in C:/Users/Dominik/Desktop/jest\_GIT/.git/

Następnym krokiem będzie określenie : **imię, nazwisko** oraz **email**.

Dominik@weirdFISH\_ MINGW64 ~/Desktop/jest\_GIT (master)

$ git config --global user.name "Dominik Beń"

Dominik@weirdFISH\_ MINGW64 ~/Desktop/jest\_GIT (master)

$ git config --global user.email "bendominik@gmail.com"

Aby sprawdzić czy dane zostały poprawnie wprowadzone korzystamy z **git config --get user.mail**

oraz **git config –get user.email**.

W repozytorium czyli de facto pustym katalogu roboczym z ukrytym katalogiem .git stworzymy niezależny plik tekstowy i wpiszemy do niego losowy ciąg znaków. Aby sprawdzić status naszych plików wpisujemy **git status** w wyniku czego otrzymujemy:

Dominik@weirdFISH\_ MINGW64 ~/Desktop/jest\_GIT (master)

$ git status

On branch master

Initial commit

Untracked files:

(use "git add <file>..." to include in what will be committed)

pierwszy\_plik.txt

nothing added to commit but untracked files present (use "git add" to track)

# Trudne słówka

• • •

Migawka – stan projektu w danej chwili. Określony kluczem SHA oraz posiadający informacje o użytkowniku dokonującym zmiany oraz dacie dokonanych zmian. Wykorzystywane do odtwarzania oraz porównywania wersji projektu w określonym migawką stanie.

Dostaliśmy informację o tym, że mamy jeden nieobserwowany plik oraz że nic aktualnie nic nie możemy zatwierdzić. Wszystkie nowe pliki w repozytorium wymagają dodatkowego śledzenia. Tak więc aby dodać do plików śledzonych nasz pierwszy\_plik.txt należy posłużyć się komendą

**git add pierwszy\_plik.txt** równie dobrze, gdy chcemy dodać wszystkie nieśledzone pliki korzystamy z **git add \***. Jeżeli chcemy dodać tylko pliki .txt: **git add \*.txt**. Nowe foldery w repozytorium nie są uwzględniane.

Aby przestać śledzić pliki konieczne jest skorzystanie z :

**git rm -r --cached pierwszy\_plik.txt** . Metoda ta umożliwia usunięcie z poczekalni wybranego pliku, nawet kiedy został już zatwierdzony. Jednak aby usunąć ze śledzenia plik zatwierdzony należy również zatwierdzić usunięcie (**commit**’y opisane są niżej).

Każdy nowododany plik znajdujący się w poczekalni w celu potwierdzenia dokonanych zmian należy dodatkowo zatwierdzić opisując co w danej migawce zostało wykonane. Zatwierdzenie odbywa się poprzez komendę **commit**:

**git commit –m „Nazwa dla migawki”**

Po dodaniu plików do poczekalni i zmodyfikowaniu go, należy podownie dodać taki plik do poczekalni poleceniem

**git add <nazwa\_pliku>**.

Pliki zatwierdzone ale zmodyfikowane nie są uwzględniane przy commitach.

Dominik@weirdFISH\_ MINGW64 ~/Desktop/jest\_GIT (master)

$ git status

On branch master

Changes to be committed:

(use "git reset HEAD <file>..." to unstage)

new file: lol.txt

new file: pierwszy\_plik.txt

new file: qwe.txt

new file: qweqwe.txt

Changes not staged for commit:

(use "git add <file>..." to update what will be committed)

(use "git checkout -- <file>..." to discard changes in working directory)

modified: qweqwe.txt

Untracked files:

(use "git add <file>..." to include in what will be committed)

qweqwe

qweqwe.c

Gdy w takim przypadku zatwierdzimy wszystkie zmiany plik qweqwe.txt, który będąc w poczekalni został zmodyfikowany nie zostanie zatwierdzony.

Po zatwierdzeniu zmian (zacommitowaniu), aby sprawdzić, które pliki aktualnie są obserwowane posługujemy się poleceniem:   
**git ls-files** .

Pliki dodane będące w poczekalni, a jeszcze nie zatwierdzone również zostaną wyświetlone.

Aby wysłać pliki do naszego serwera posłużymy się poleceniem

Dominik@weirdFISH\_ MINGW64 ~/Repozytorium (master)

$ git remote add origin <https://github.com/Wildandwest/hello_world.git>

Do zweryfikowania nowego miejsca zarządzalnego wpisujemy

Dominik@weirdFISH\_ MINGW64 ~/Repozytorium (master)

$ git remote -v

W przypadku gdy chcemy nadać nowy adres. . .:

Dominik@weirdFISH\_ MINGW64 ~/Repozytorium (master)

$ git remote set-url origin https://github.com/Wildandwest/hello\_world.git