GIT

1.1 Początki z systemem kontroli wersji (GIT)

W wierszu poleceń sprawdź, czy masz zainstalowanego GITa:

```
git  --version
```

Na systemach UNIXowych:

```
which git
```

Przykładowe poprawne wyjście:

```
git version 2.16.1.windows.1
```

Jeśli nic się nie pojawia, oznacza to, że GIT nie jest zainstalowany na Twoim komputerze.
Link do pobrania: https://git-scm.com

1.2 Ustawienie adresu email i nazwy użytkownika

Musisz ustawić kim jesteś, przed stworzeniem **commita**. Dzięki temu każdy commit będzie poprawnie podpisany Twoim adresem email i nazwą.

By zadeklarować swoją tożsamość do wszystkich repozytoriów, użyj git config -- global
Zostanie to przechowane w ustawieniach Twojego użytkownika w pliku .gitconfig
Np. %USERPROFILE%\.gitconfig (Windows) lub $HOME/.gitconfig (UNIX)

```
git config --global user.name "Twoja Nazwa"
git config --global user.email mail@example.com
```

By zadeklarować tożsamość dla pojedynczego repozytorium, użyj git config będąc w lokalizacji repozytorium.

```
cd \sciezka\do\mojego\repozytorium
git config user.name "Twoj Login w Pracy"
git config user.email mail_praca@example.com
```

Gdy nie ustawisz nazwy użytkownika i emaila dla repozytorium i będziesz próbował zrobić **commit**, zobaczysz taki błąd:

```
no name was given and auto-detection is disabled
no email was given and auto-detection is disabled
```

1.3 Tworzenie pierwszego repozytorium

Gdy Git jest już zainstalowany, przejdź do folderu w którym chcesz utworzyć repozytorium, użyj komendy:

```
git init
```

Komenda tworzy ukryty folder .git, który zawiera wszystko, czego potrzebujesz do pracy z gitem.

By sprawdzić listę plików, które będą commitowane, użyj komendy:

```
git status
```

Po wyświetleniu listy, możesz powiedzieć Gitowi, które pliki mają być kontrolowane pod względem zmian. Służy do tego komenda:

```
git add plik/ścieżka-do-pliku    plik2/ścieżka-do-pliku2 itd.
```

Jeśli wszystkie pliki w folderze mają być dodane do repozytorium, użyj komendy:

```
git add .
```

By wysłać pliki do repozytorium wraz z komentarzem, użyj komendy:

```
git commit -m "Komentarz"
```

Stworzy ona nowy commit z przekazanym komentarzem. Commit jest jak zapis Twojego całego projektu. Teraz możesz użyć komendy push aby zaktualizować swoje repozytorium. Później możesz powrócić do poprzedniej wersji pliku, jeśli byłoby to konieczne. Jeśli pominiesz parametr -m, zostanie otwarty wybrany w trakcie instalacji edytor tekstu (domyślnie VIM).

Dodawanie linku do repozytorium (remote)

Aby dodać nowy link, użyj komendy `git remote add` w folderze, w którym przechowujesz swoje repozytorium.

Komenda `git remote add` przyjmuje dwa argumenty:

1. Remote name, np. **origin**
2. Remote URL, np. https://<adres-twojego-serwisu-git>/uzytkownik/repozytorium.git

```
git remote add origin
https://<adres-twojego-serwisu-git>/uzytkownik/repozytorium.git
```

UWAGA: Przed dodaniem linku, musisz stworzyć repozytorium w Twoim serwisie git, wtedy będziesz mógł używać komend push/pull.

Wysyłanie plików do repozytorium

Podczas pierwszego wysyłania plików należy użyć komendy `git push` z argumentem `--set-upstream`.

```
git push --set-upstream origin master
```

Przy każdym następnym wysyłaniu plików, wystarczy bezargumentowa komenda:

```
git push
```

1.4 Klonowanie repozytorium

Aby pobrać kopię repozytorium należy wejść do folderu, w którym ma ono się znajdować, a następnie użyć komendy `git clone`.

Przykładowe użycie dla GitHub'a:

```
cd <ścieżka do folderu>
git clone https://github.com/nazwa_użytkownika/nazwa_repozytorium.git
```

Polecenie tworzy folder o nazwie `nazwa_repozytorium` zawierający wszystkie pliki znajdujące się w repozytorium. Dodany zostanie również folder .git, który zawiera całą historię i konfigurację.

Aby nadać własną nazwę należy ją dopisać za adresem repozytorium:

```
git clone
https://github.com/nazwa_użytkownika/nazwa_repozytorium.git mojaNazwa
```

Jeśli chcemy sklonować do obecnego folderu należy użyć kropki:

```
git clone https://github.com/nazwa_użytkownika/nazwa_repozytorium.git .
```

1.5 Wyświetlanie informacji o komendzie

Aby wyświetlić informacji o jakiejkolwiek komendzie gita należy użyć argumentu `--help` lub przed komendą napisać `help`:

```
git diff --help
git help diff
```

1.6 Zmiany pomiędzy dwoma commitami

```
git diff commit1..commit2
```

Ta komenda pokaże tekstowe różnice pomiędzy dwoma commitami, niezależnie od tego gdzie znajdują się w hierarchii.

Jeśli chcesz porównać obecny plik, z jego poprzednią wersją, użyj tej komendy:

```
git diff @~1..@
```

Cyfra po tyldzie oznacza kolejną wersję - jeśli chcesz porównać 3 wersje wstecz, wpisz 3, itd.

1.7 Commit

1.7.1 Podstawy

Po przeprowadzeniu zmian w kodzie, powinieneś pogrupować zmiany w Gicie przed przeprowadzeniem commita.
Np. jeśli zmienisz plik README.md i program.cpp:

```
git add README.md program.cpp
```

Tą komendą informujesz gita, jakie pliki dodać do następnego commita.
Później commitujesz swoje zmiany za pomocą:

```
git commit
```

Pamiętaj, że ta komenda otworzy edytor tekstu (domyślnie vim). Jeśli nie miałeś styczności z Vim'em, powinieneś wiedzieć, że

- "**i**" otwiera tryb pisania (*insert mode*)
- teraz możesz wpisać komentarz commita
- aby zakończyć edycję, wciśnij `Esc` i wpisz `:wq` by zapisać i wyjść.

Jeśli chcesz uniknąć wchodzenia w edytor tekstu, dodaj do komendy znacznik `-m` z Twoją wiadomością commita.

```
git commit -m "zawartość komentarza"
```

Jeśli zmieniłeś wiele plików w katalogu możesz dodać znacznik `--all` zamiast wymieniać każdy plik z osobna.

```
git add --all
```

Jeśli chcesz dodać wszystkie pliki znajdujące się w obecnym folderze i podfolderach, użyj:

```
git add .
```

Gdy chcesz dodać tylko pliki, które są już ‘śledzone’ i zostały zaktualizowane użyj komendy:

```
git add -u
```

Jeśli chcesz podejrzeć wprowadzone zmiany:

```
git status
git diff --cached
```

Jeśli tylko zmodyfikowałeś albo usunąłeś pliki i nie stworzyłeś żadnych nowych, możesz połączyć komendy `git add` i `git commit` w jedną:

```
git commit -am "Wiadomość commita"
```

Ta komenda doda wszystkie zmodyfikowane pliki tak samo jak `git add --all`.

1.7.2 Jak pisać dobre wiadomości do commita

Siedem zasad dobrej wiadomości:

1. Oddziel tytuł od treści linią przerwy.
2. Ogranicz tytuł do 50 znaków
3. Tytuł pisz dużymi literami
4. Nie kończ wiadomości kropką
5. Tytuł powinien być pisany w formie komendy
6. Ręcznie dodawaj nowe linie
7. Używaj treści, aby wyjaśnić ***co?*** i ***dlaczego?*** zamiast ***jak?***

7 zasad z *Chris Beam’s blog. https://chris.beams.io*

Parę przykładów dobrych wiadomości:

```
TASK-123:   Zaimplementowanie logowania przez 0Auth
TASK-124:   Dodanie zmniejszania rozmiaru w plikach JS/CSS
TASK-125:   Naprawienie błędu zmniejszania pliku
```

Parę przykładów złych wiadomości:

```
naprawa                                  //Co zostało naprawione?
trochę zmian                          //Zmian czego?
TASK-371                                  //Brak opisu, czytający musi
sprawdzić co to za zadanie, aby zrozumieć
Zaimplementowany "IFoo"  w "IBar"         //Po co było to potrzebne?
```

1.8 Pobieranie zmian z serwera do lokalnego repozytorium (PULL)

Gdy pracujesz w zespole na zdalnym repozytorium (np. GitHub), chcesz w jakimś stopniu dzielić zmiany razem z nimi. Gdy twoi koledzy wyślą (push) zmiany do zdalnego repozytorium, możesz pobrać je za pomocą komendy:

```
git pull
```