Git

repozytorium lokalne

Czego się dowiesz?

- Czym jest repozytorium lokalne?
- Jak je stworzyć?
- Jak zapisywać zmiany, tworzyć historię kodu?
- Jak niezależnie tworzyć kod na różnych "gałęziach"?

Czym jest?

Repozytorium lokalne, to takie repozytorium (miejsce do przechowywania kodu), które jest zlokalizowane na Twoim komputerze.

Konfiguracja

Zanim rozpoczniesz pracę w gitem, ustaw informacje o użytkowniku, który z niego korzysta.

Otwórz konsolę Git Bash lub systemową, inaczej nazywaną "Wiersz polecenia" lub "Terminal".

Wprowadź komendy, wypełniając swoimi danymi:
git config --global user.email "twój email"
git config --global user.name "twoje imię i nazwisko"

Pierwsze repozytorium

Repozytorium kodu możesz stworzyć w dowolnym folderze na komputerze.

- stwórz nowy katalog "projekt_git"
- wewnątrz katalogu kliknij prawym przyciskiem myszy i wybierz Git Bash Here
- wpisz w konsoli komendę: **git init**

```
leyas@DESKTOP-N70HSFF MINGW64 /d/APJ/git/projekt_git
$ git init
Initialized empty Git repository in D:/APJ/git/projekt_git/.git/
```

W ten sposób stworzyliśmy pierwsze repozytorium lokalne.

Status repozytorium

W dowolnym momencie pracy z kodem, możemy sprawdzić jego status.

- wpisz w konsoli komendę: **git status**

```
leyas@DESKTOP-N7OHSFF MINGW64 /d/APJ/git/projekt_git (master)
$ git status
On branch master

No commits yet

nothing to commit (create/copy files and use "git add" to track)
```

Dowiadujemy się, że jesteśmy na branchu "master", że nie ma żadnych "commitów" oraz jeśli chcemy coś zapisać to powinniśmy najpierw wywołać komendę "git add". Wszystkie te terminy, omówię na kolejnych slajdach.

Zapisywanie zmian

Od momentu stworzenie repozytorium katalog jest monitorowany i wszelkie zmiany w plikach są znane. Nie oznacza to jednak, że są automatycznie zapisywane w repo.

Wykonaj kroki:

- stwórz plik o nazwie imiona.txt, wewnątrz dodaj kilka imion i zapisz
- dodaj plik do tzw. tracka, a więc do śledzenia przez Gita

  git add imiona.txt
- zapisz zmianę opisując co zostało wykonane, np.:

  git commit -m "dodanie pliku z imionami"

Przełącznik -m oznacza "message", a więc wiadomość opisującą zmianę.

```
leyas@DESKTOP-N70HSFF MINGW64 /d/APJ/git/projekt_git (master)
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        imiona.txt

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

leyas@DESKTOP-N70HSFF MINGW64 /d/APJ/git/projekt_git (master)
$ git add imiona.txt

leyas@DESKTOP-N70HSFF MINGW64 /d/APJ/git/projekt_git (master)
$ git commit -m "dodanie pliku z imionami"
[master (root-commit) ee0e8be] dodanie pliku z imionami
 1 file changed, 1 insertion(+)
 create mode 100644 imiona.txt
```

Zauważ, że przed wykonaniem: git add oraz git commit, wykonałem też git status, które poinformowało mnie, że pojawił się nowy plik do zapisania.

Historia zmian

Aby podejrzeć, co do tej pory zostało zapisane w repozytorium, wykonamy komendę: **git log**

```
leyas@DESKTOP-N70HSFF MINGW64 /d/APJ/git/projekt_git (master)
$ git log
commit ee0e8beec5b7ba379719f61bb188800bc99c9bb5 (HEAD -> master)
Author: Szymon Leyk <leyas@wp.pl>
Date:   Mon Nov 23 20:08:15 2020 +0100

    dodanie pliku z imionami
```

Jak widzisz, commit otrzymał unikalny numer (tzw. hash), który jest jego identyfikatorem. Zostały dodane informacje o autorze oraz data zapisu.

Zadanie 1

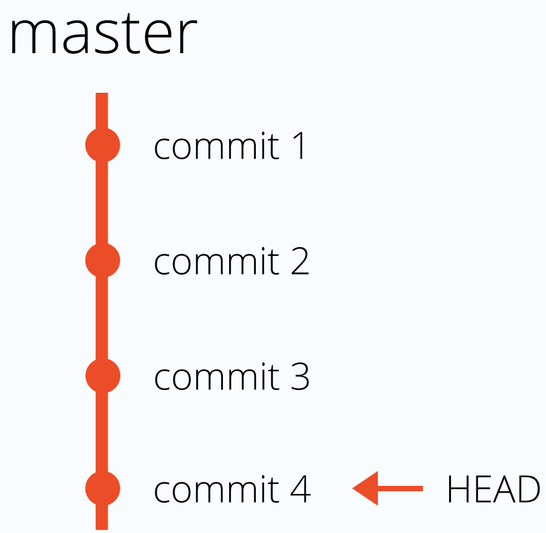
Stwórz kilka zmian:

- dodaj nowe imiona do pliku imiona.txt
- sprawdź status repozytorium
- wykonaj komendy git add oraz git commit
- sprawdź status repozytorium
- dodaj nowy plik nazwiska.txt, dodaj nazwiska i zapisz
- sprawdź status repozytorium
- wykonaj komendy git add oraz git commit
- sprawdź status repozytorium
- wykonaj komendę git log

Branch - czym jest?

branch - gałąź - jest miejscem, na którym zapisywane są zmiany. Domyślnie, każde repozytorium otrzymuje branch master. Każdy wykonany do tej pory commit lądował na gałęzi master.

HEAD - wskazuje na gałąź, na której się znajdujesz.



Branch - po co?

Jest jeden główny powód: chęć rozwijania, tworzenia, poprawiania kodu w oderwaniu od obecnego.
Gdy masz działający kod, to ostatnią rzeczą, której chcesz to utrata jego działania.

Idealnym rozwiązaniem jest utworzenie nowej gałęzi. Tak więc, jeśli masz kod, który działa i chcesz dodać do niego nową funkcjonalność, to tworzysz nowy branch i na nim pracujesz tak długo, dopóki funkcjonalność nie zostanie ukończona.

Gdy nowa funkcjonalność będzie już działała, scalasz ją ponownie z branchem z którego "wyszedłeś".

Branch - jak?

Będąc na branchu master wykonaj komendę:
git checkout -b zadanie1

Komenda ta stworzy nowy branch, który będzie posiadał wszystko co jest na masterze. Nazwa nowego brancha, to "zadanie1".

Od teraz możesz rozwijać kod, nie wpływając na ten, który jest na masterze.

Scalanie branchy

Ukończyłeś już zadanie 1? Aby scalić branch zadanie1 z branchem master, wykonaj kroki:

- przejdź na branch master: **git checkout master**
- połącz z gałęzią zadanie1: **git merge zadanie1**

Zauważ, że przełącznik -k w komendzie git checkout, występuje tylko wtedy, gdy tworzymy nowy branch. W przeciwnym wypadku komenda checkout służy nam do przechodzenia między branchami.

Sprawdź branche

Do sprawdzania, jakie branche posiadamy w repozytorium, służy komenda:
git branch

Branch, przy którym znajdziesz oznaczenie *, to ten na którym jesteś.

Podsumowanie

git init - stworzenie repozytorium
git status - sprawdzenie statusu repozytorium
git add nazwa_pliku - dodanie pliku do "śledzenia"
git commit -m "opis" - zapisanie zamiany
git checkout -b nazwa_brancha - stworzenie nowego brancha
git checkout nazwa_brancha - przejście na inny branch (już istniejący)
git merge nazwa_brancha - scalenie brancha o podanej nazwie z tym, na którym jesteśmy

Dodatkowe materiały

Wprowadzenie do Gita - tutorial po polsku
Ściągawka z komendami od JRebel

Zadanie 2

- stwórz nowy katalog o nazwie notatnik
- stwórz w tym katalogu repozytorium gita
- na branchu master dodaj plik "co_mam_do_zrobienia.txt", wypełnij danymi
- zapisz zmiany
- stwórz nowy branch o nazwie "plany"
- na branchu plany dodaj kilka zmian (kilka commitów), dodając za każdym razem jakiś plan na przyszły rok w pliku "co_mam_do_zrobienia.txt"
- po dodaniu kilku commitów, scal branch "plany" z branchem master
- wywołaj: git log