Git Ściąga

Konfiguracja

Ustawienie nazwy użytkownika

```
git config --global user.name „Jan Kowalski"
```

Ustawienie maila użytkownika

```
git config --global user.email jankowalski@gmail.com
```

Wyświetlenie danych użytkownika

```
git config --list
```

Tworzenie Repozytorium

Inicjalizacja Repozytorium Lokalnego

```
git init
```

Klonowanie Repozytorium Zdalnego

```
git clone <url>
```

Dodawanie Zmian

Wyświetlanie zmian w katalogu roboczym i poczekalni

```
git status
```

Podgląd dokonanych zmian

```
git diff
```

Dodawanie plików do poczekalni

```
git add <plik1> <plik2>
```

Dodawanie wszystkich nowych i zmodyfikowanych plików do poczekalni

```
git add .
```

Dodawanie wszystkich zmian plików do poczekalni

```
git add -A
```

Zapisywanie w repozytorium wszystkich zmian z poczekalni

```
git commit -m „opis zmian"
```

Dodanie do poczekalni wszystkich zmian i zapis w repozytorium

```
git commit -am „opis"
```

Cofanie i usuwanie zmian

Usuwanie pliku z projektu i zapisanie w poczekalni informacji o jego usunięciu

```
git rm <plik1>
```

Usuwanie wszystkich wprowadzonych zmian w pliku z katalogu roboczego

```
git checkout -- <plik1>
```

Usuwanie wszystkich wprowadzonych zmian we wszystkich plikach katalogu roboczego

```
git checkout --
```

Usuwanie pliku z poczekalni

```
git rm --cached <plik1>
```

Cofanie wybranego commita poprzez zatwierdzenie jego przeciwieństwa

```
git revert <commit-id>
```

Usuwanie ostatniego commita ale z zachowaniem zmian w katalogu roboczym

```
git reset HEAD^
```

Usuwanie wszystkich wprowadzonych zmian w katalogu roboczym, poczekalni, repozytorium i powrót do stanu projektu z ostatniego commita

BĄDŹ OSTROŻNY

```
git reset --hard HEAD^
```

Poprawianie commita

```
git commit --amend
```

Branche

Pokazywanie wszystkich branchy z repozytorium lokalnego

```
git branch
```

Pokazywanie wszystkich branchy z repozytorium lokalnego i zdalnego

```
git branch -a
```

Tworzenie nowego brancha na podstawie HEAD

```
git branch <nazwa>
```

Tworzenie nowego brancha na podstawie wybranego commita

```
git branch <nazwa> <commit-id>
```

Tworzenie nowego brancha i przełączenie się na niego

```
git checkout -b <nazwa>
```

Przełączanie się na wybranego brancha

```
git checkout <nazwa>
```

Usuwanie brancha z repozytorium lokalnego

```
git branch -d <nazwa>
```

Dodawanie taga do obecnego commita

```
git tag <nazwa>
```

Budowanie relacji pomiędzy lokalnym i zdalnym branchem

```
git branch --set-upstream-to=origin/<nazwa-zdalnego-brancha> <nazwa-lokalnego-brancha>
```

Wyświetlanie zmian pomiędzy dwoma branchami

```
git diff branch1..branch2
```

Wyświetlanie historii commitów z wybranego brancha

```
git log
```

Wyświetlanie kto co zmienił w wybranym pliku

```
git blame <plik1>
```

Wyświetlanie historii commitów i przełączania się pomiędzy branchami

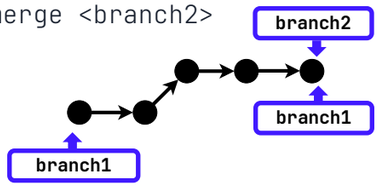
```
git reflog
```

Merge & Rebase

Dołączanie innego brancha do tego, na którym jesteśmy

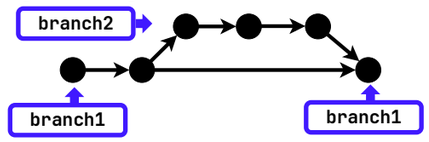
Metoda domyślna, gdy branch1 pozostał bez zmian, commitowaliśmy tylko na branch2 (fast-forward)

```
git checkout <branch1>
git merge <branch2>
```

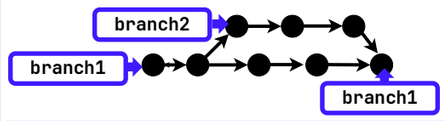


Metoda no fast forward, stosowana w celu zachowania przejrzystej historii zmian

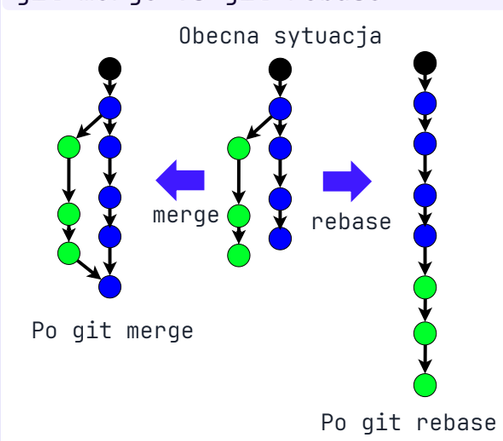
```
git merge <branch2> --no-ff
```



Metoda domyślna, gdy branch1 i branch2 ulegały zmianie ale nie modyfikowano tych samych plików (brak konfliktów)



git merge vs git rebase



Przerwanie scalania metodą rebase w przypadku konfliktów

```
git rebase --abort
```

Kontynuacja scalania po rozwiązaniu konfliktów

```
git rebase --continue
```

Złota zasada scalania zmian za pomocą git rebase

Nigdy nie używaj git rebase na branchach publicznych ponieważ możesz doprowadzić do poważnych problemów

Repozytorium Zdalne

Wyświetlanie informacji o repozytoriach zdalnych

```
git remote -v
```

Wyświetlanie informacji o wybranym repozytorium zdalnym

```
git remote show <zdalne>
```

Dodawanie nowego repozytorium zdalnego

```
git remote add <zdalne> <url>
```

Pobieranie zmian ze zdalnego repo- bez łączenia z plikami lokalnymi

```
git fetch <zdalne>
```

Pobieranie zmian z repo zdalnego (z połączonego brancha) + łączenie z plikami lokalnymi

```
git pull
```

Domyślne git pull używa git merge, jest jednak możliwość skonfigurowania go tak, aby używał git rebase (zalecana opcja).

```
git config --global pull.rebase true
```

Wysyłanie zmian z repo lokalnego do zdalnego

```
git push
```

Następnie możesz wykonać Pull Request (PR)

Usuwanie brancha w zdalnym repo

```
git push origin -d branch-name
```

Schowek

Odkładanie niezapisanych zmian w schowku, gdy musisz zmienić branch

Gdy nie chcesz robić commita ale musisz przełączyć się na inny branch A jednocześnie nie chcesz utracić zmian

```
git stash
```

Pokazanie ostatnio dodanych plików do schowka

```
git stash show
```

Przywrócenie ostatnio dodanego kodu ze stasha do katalogu roboczego

```
git stash apply
```

Praca z Gitem

1

Tworzenie nowego projektu

`git init`

Wykonanie tej komendy w głównym folderze projektu tworzy nowe repozytorium gita (w folderze .git) - od teraz korzystasz z systemu kontroli wersji

Klonowanie repozytorium

`git clone <repo-url>`

Wykonanie tej komendy w głównym folderze projektu pobiera kopię repozytorium zdalnego np. z Githuba. Od teraz masz własną kopię projektu z jego całą historią i wszystkimi plikami

2

Praca na własnych plikach (lokalnie)

Od teraz masz pełną swobodę i modyfikujesz pliki tak jak chcesz i w wybranym edytorze kodu źródłowego. Dzięki temu, że robisz to na własnej kopii projektu, nie możesz nic popsuć (na produkcji).

3

Co się zmieniło

`git status`

To polecenie zwraca informację co zmieniło się od ostatniego commita.

4

Dodawanie plików do Poczekalni

`git add <nazwa-pliku>`

Każdy zmodyfikowany plik musi być dodany do Poczekalni jeśli chcesz uwzględnić te zmiany w kolejnym commicie.

5

Zatwierdzenie zmian (Wykonanie commita)

`git commit -m „opis"`

Zebranie wszystkich dotychczasowych zmian zapisanych w poczekalni i zapisanie ich w bazie danych Gita. Inaczej nazywa się to zapisaniem stanu projektu.

6

Sprawdzanie stanu projektu

`git status`

Częste sprawdzanie statusu katalogu roboczego i poczekalni to dobry nawyk a dodatkowo wykonanie tego polecenia po commicie udowodni, że tylko pliki znajdujące się w poczekalni zostały uwzględnione w commicie.

7

Wyświetlanie historii commitów

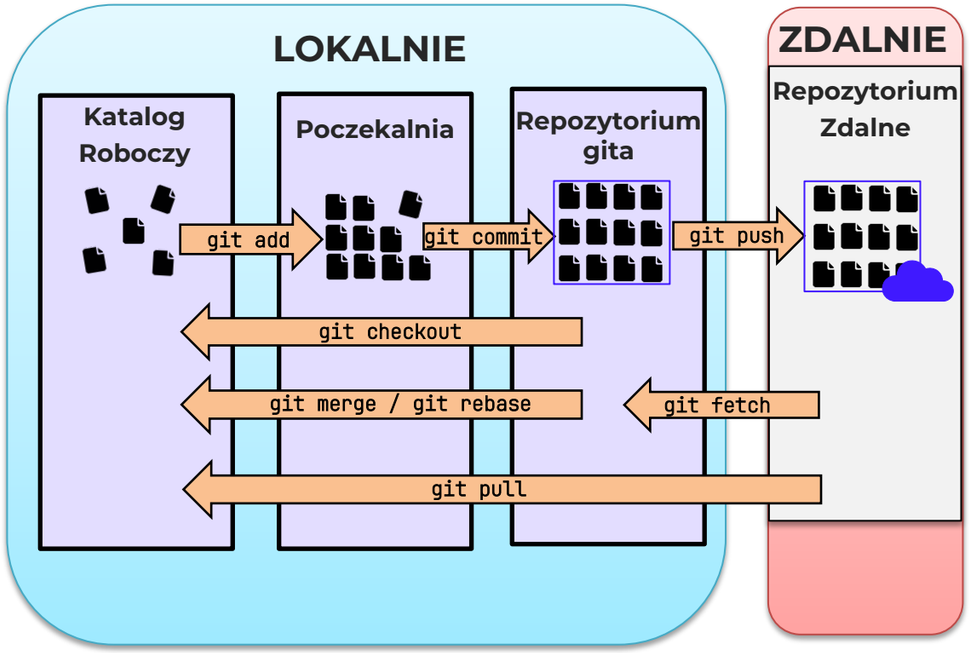
`git log`

Wyświetlenie wszystkich wykonanych commitów w kolejności chronologicznej. Pozwala zorientować się jak wygląda historia zmian projektu.

W.PRUCHNICKI

git AKADEMIA

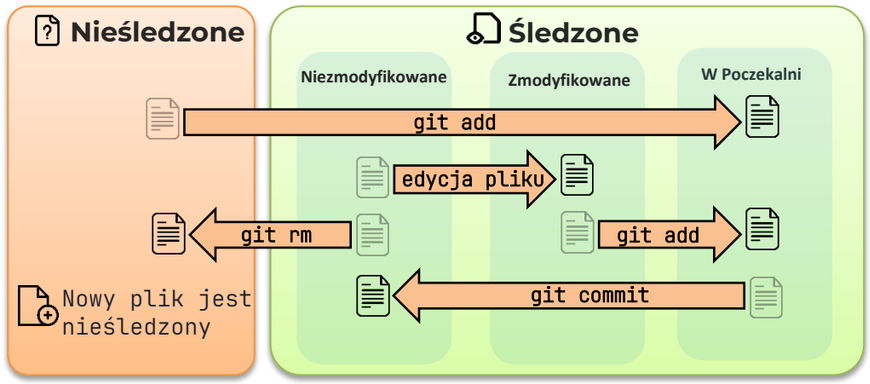
Model Pracy z Gitem



Cykl życia stanu pliku

Pliki, o których git nie ma informacji (wie tylko, że istnieją)

Pliki o których git wie i ma o nich szczegółowe informacje



Dobre Praktyki

Commity powinny być małe i robione często

Jeśli dokonasz zmiany, która działa i ma sens, nawet gdy jest mała to rób commita. Częste commity pozwalają na stosowanie krótkich opisów i zapewniają spójną historię zmian.

Zadbaj o dobre opisy commitów

Jest to jeden z najważniejszych a zarazem najczęściej lekceważonych aspektów kontroli wersji. Opis commitu ma mieć sens i przekazywać konkretne informacje żeby pomóc Tobie i innym zrozumieć co się zmieniło.

Używaj branch-y

Gdy pracujesz nad projektem to warto korzystać z branchy. Szczególnie przydatne jest to w trakcie pracy z innymi gdy pracując nad nową funkcjonalnością tworzysz nowy branch i po zakończeniu zadania robisz Pull Request aby dołączyć zmiany do głównego brancha. Pamiętaj One Feature - One branch.

Nie uwzględniaj w commicie niepotrzebnych i wrażliwych danych

Pliki konfiguracyjne zawierające klucze, hasła itd. oraz pliki i foldery zawierające zależności (np. w js node_modules) nie powinny być uwzględniane w commitach. Do ignorowania takich plików przez Gita służy plik .gitignore

Nie commituj niedziałającego kodu

Zanim wykonasz commit upewnij się, że kod działa i chcesz zatwierdzić uwzględnione zmiany. Gdy musisz coś zmodyfikować/sprawdzić użyj git stash ale commituj tylko to co działa prawidłowo i zostało przez Ciebie przetestowane.

Różne nazwy takie samo znaczenie

Katalog roboczy, working directory	**Poczekalnia, Przechowalnia, Staging area, Index**