Systemy kontroli wersji + Cygwin

Michał Orlikowski
michalorlik@gmail.com [SDA]

System kontroli wersji

System kontroli wersji (ang. version/revision control system) – oprogramowanie służące do śledzenia zmian głównie w kodzie źródłowym oraz pomocy programistom w łączeniu zmian dokonanych w plikach przez wiele osób w różnych momentach czasowych.

Cygwin/MinGW

Środowisko, zestaw narzędzi, które udostępniających pod Windowsem podobna funkcjonalność co system Linux.

Cygwin/MinGW

Najważniejsze polecenia
http://www.cheat-sheets.org/saved-copy/linux-command-line-cheat-sheet.png

Więcej:

http://www.ee.surrey.ac.uk/Teaching/Unix/index.html

Krótki przegląd systemów kontroli wersji

- Lokalne (foldery, pliki, kopie, RCS)
- Scentralizowane (Subversion, CVS, Perforce, ClearCase)
- Rozproszone (BitKeeper, Baazar, **Git**, Mercurial)

Skrócona historia Gita

2002 r. - projekt jądra Linuxa zaczyna używać BitKeepera
2005 r. - BitKeeper stał się płatny. Spór. Pomysł stworzenia własnego bezpłatnego systemu kontroli wersji. Cele:

- Szybki
- Prosty
- Rozproszony
- Silnie nastawiony na równoległe rozwijanie wielu różnych gałęzi
- Musi być w stanie obsłużyć duże projekty (jak jądro Linuxa)

Po 4 dniach git był w stanie przechowywać swój własny kod.
Kwiecień-Lipiec wypuszczenie wersji 0.99 (aktualnie 2.11.0)

Czym jest GIT?

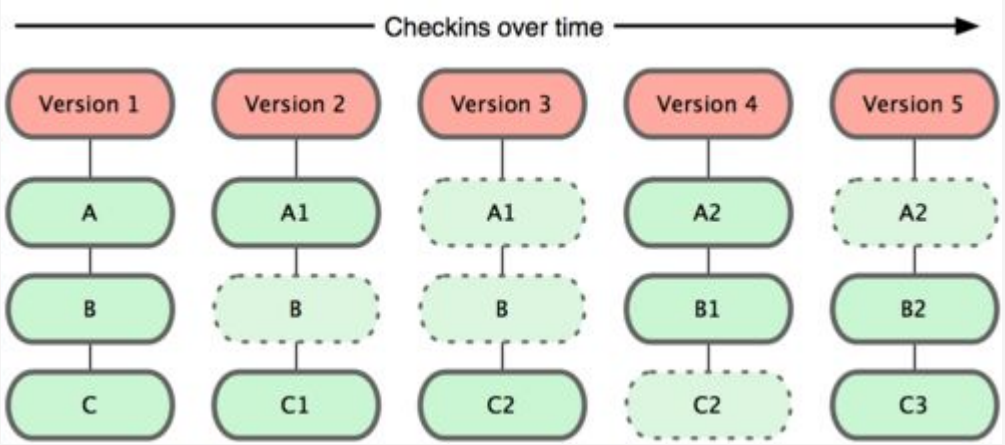
Git to szybki, rozszerzalny, rozproszony system kontroli wersji z bardzo bogatą listą komend. Cechy gita:

- Przechowuje migawki nie różnice
- Prawie wszystkie operacje są lokalne
- Wbudowane mechanizmy spójności danych
- GIT z reguły dodaje tylko nowe dane (są wyjątki!)

Migawki a różnice

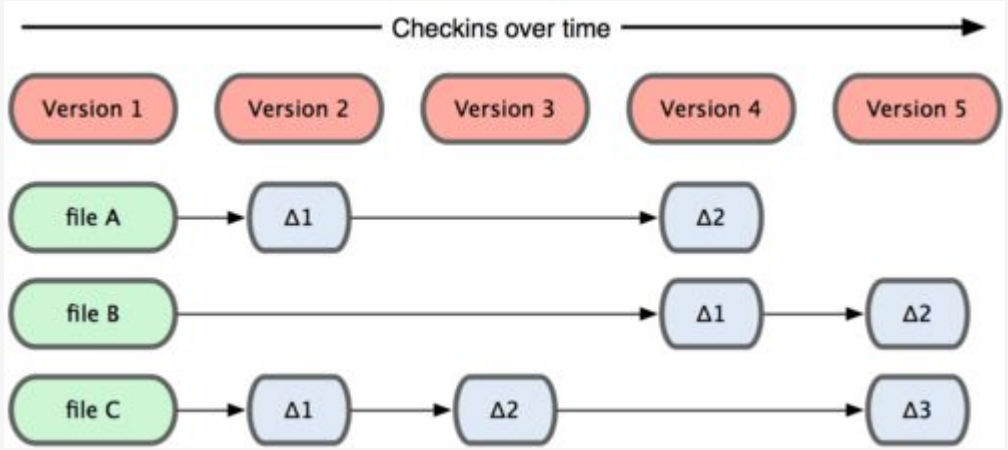
Migawki [snapshot]

"Kopia" plików w danym momencie



Różnice [delta]

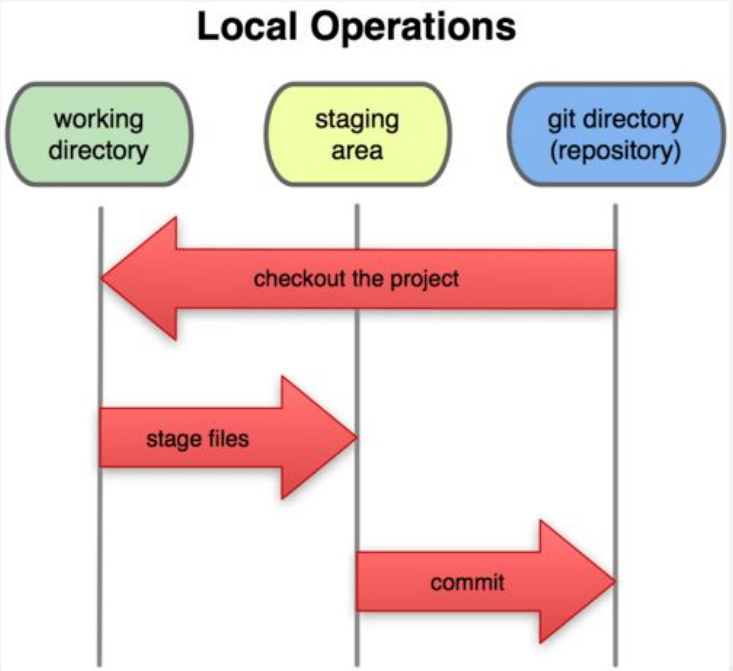
Lista zmian na plikach

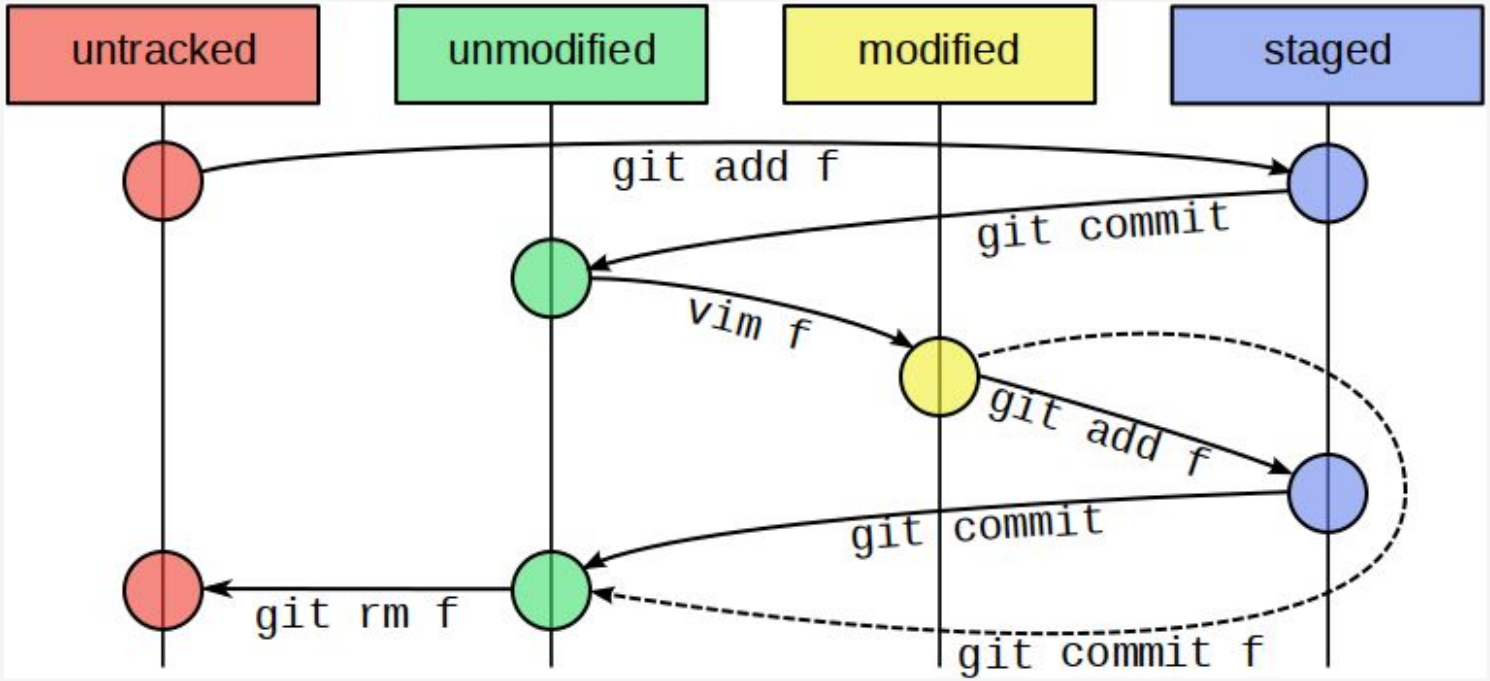


Źródło: https://git-scm.com/book/pl/v1/Pierwsze-kroki-Podstawy-Git

Stany plików

1. Nieśledzony (jest poza Gitem)
2. Niezmodyfikowany (taki jak w repozytorium)
3. Zmodyfikowany
4. Przechowywany (ale nie zcommitowany)





Źródło: https://github.com/git/git-scm.com/issues/288 ; https://git-scm.com/book/pl/v1/Pierwsze-kroki-Podstawy-Git

Dlaczego GIT?

- Najszybszy
- Najmniejszy
- Daje dużo możliwości
- Umożliwia pracę offline
- Choć nie jest idealny (brak możliwości zarządzania uprawnieniami wewnątrz repozytorium, “trudniejszy” niż SVN)

Instalacja

Strona projektu: https://git-scm.com/downloads

1. Pobrać wersję dla swojego systemu
2. Zainstalować
3. Gotowe!

BitBucket

BitBucket [https://bitbucket.org/] to usługa umożliwiająca przechowywanie repozytorium git na zdalnym serwerze.

Alternatywy:

- https://gitlab.com/
- https://github.com/

Szukanie pomocy

Do wyświetlanie szczegółów polecenia służy komenda:

$ git help <command>

$ git <command> --help

W razie kiedy nie wiemy co zrobić:

– Google: git + jak coś zrobić

Ustawienie GITa

git config

- --system (per komputer)
- --global (per użytkownik)
- --local (per repozytorium)

$ git config user.name "Michal Orlikowski"

$ git config user.email "michalorlik@gmail.com"

$ git config --list

$ git config --unset

Tworzenie repo, czyli armia klonów

- Nowe repozytorium:

$ git init
może być użyty w istniejącym folderze zawierającym pliki

- Istniejące repozytorium:

$ git clone <zdalne repozytorium> <folder docelowy>
<folder docelowy> musi istnieć

Tworzenie repo, czyli armia klonów

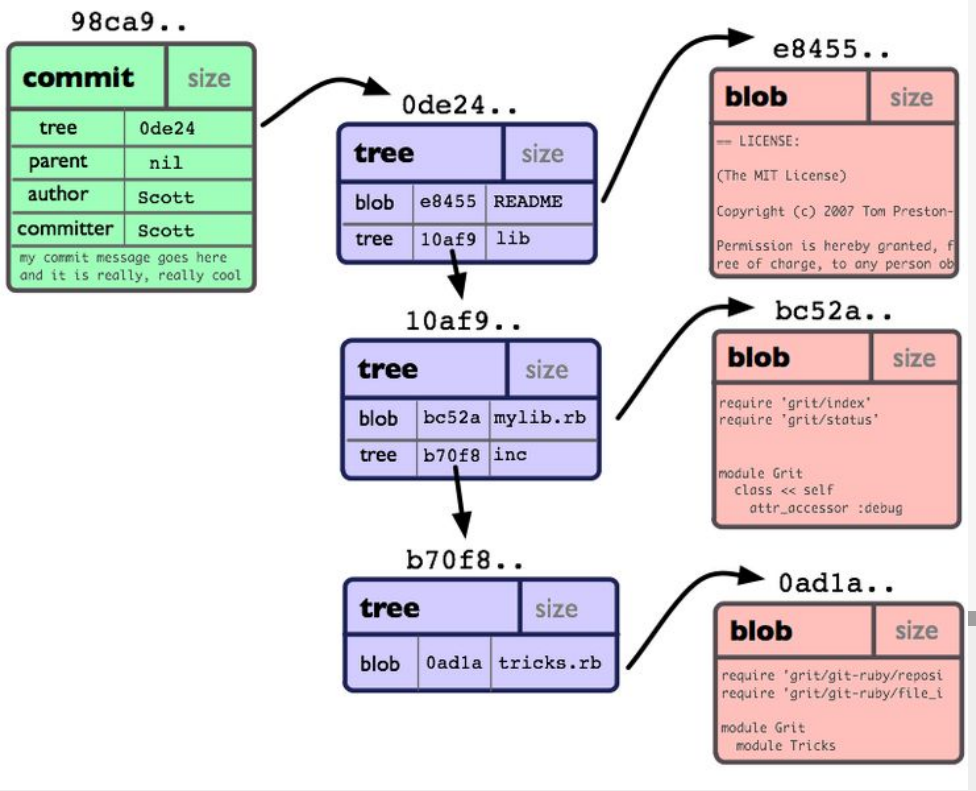
```
$ git clone https://<LOGIN>@bitbucket.org/<LOGIN>/<NAZWA_REPO>.git
$ cd <NAZWA_REPO>
$ echo "# My project's README" >> README.md
$ git add README.md
$ git commit -m "Initial commit"
$ git push -u origin master
```

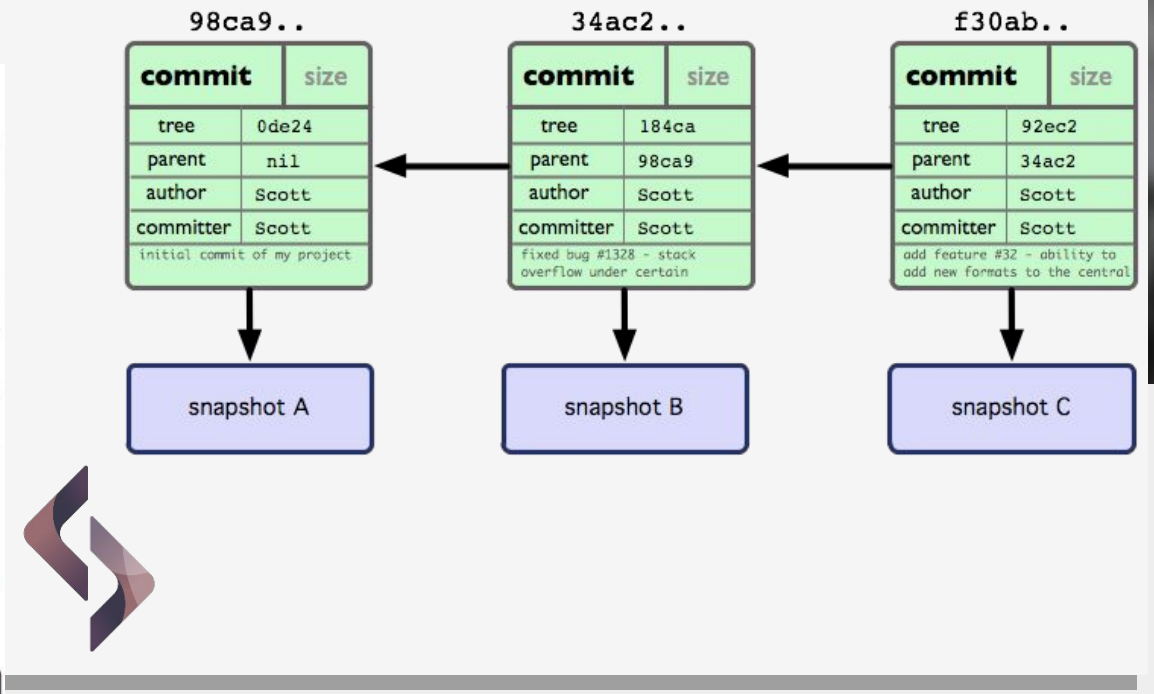
Pierwszy commit za wami

Ale jak to się stało?

Commit

Commit to zatwierdzenie zmian.





https://git-scm.com/book/pl/v1/Ga%C5%82%C4%99zie-Gita-Czym-jest-ga%C5%82%C4%85%C5%BA

Git commit

Aby zatwierdzić wszystkie zmodyfikowane pliki:

$ git commit -a -m "Opis commita"

Wybiórczo: najpierw do przechowalni, potem zatwierdzić

1. *$ git add <PLIK> | <FOLDER>*
2. *$ git commit -m "message"*

Sprawdzanie stanu 1/2

Aby sprawdzić listę commitów:

$ git log

$ git log -n lub *$ git log -*

$ git log -p

Komenda ta może być rozbudowana:

$ git log --graph --oneline --all

Sprawdzanie stanu 2/2

Aby sprawdzić stan folderu roboczego i przechowalni:

$ git status

$ git status -s [lub --short]

$ git status --long

Git add

Do dodawania plików do śledzenia/przechowalni służy komenda:

$ git add

Aby dodać wszystko

$ git add .

Można używać znaków wieloznaczności (wildcards)

*$ git add *.h*

Git rm

Do usuwania plików służy komenda

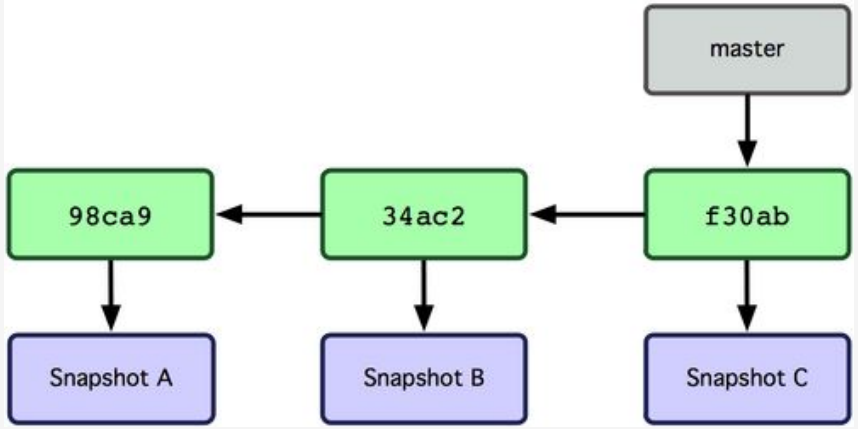
$ git rm <PLIK>

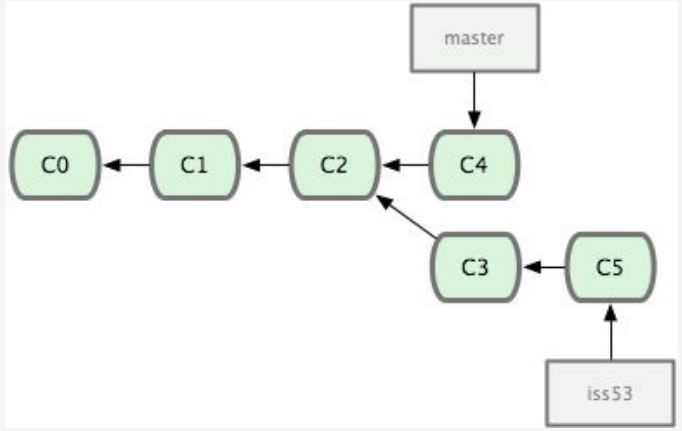
Można usunąć plik tylko z gita, pozostawiając plik na dysku

$ git rm --cached

Gałęzie

Gałąź w Gicie jest po prostu lekkim, przesuwalnym wskaźnikiem na któryś z owych zestawów zmian(commit). Domyślna nazwa gałęzi Gita to **master**.





https://git-scm.com/book/pl/v1/Ga%C5%82%C4%99zie-Gita-Czym-jest-ga%C5%82%C4%85%C5%BA

Tworzenie gałęzi

Do tworzenia gałęzi służy komenda:

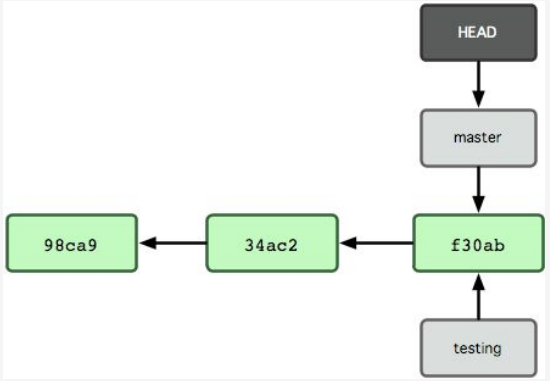
$ git branch <NAZWA> [id_commitu]

Tworzy gałąź o nazwie “nazwa” od commitu o podanym id

Bez argumentów komenda wyświetli wszystkie gałęzie

$ git branch

Gdy nie poda się id_commitu, zakłada się **HEAD** (*$ git branch testing*)



HEAD - “głowa”, najnowszy commit aktualnej gałęzi

https://git-scm.com/book/pl/v1/Ga%C5%82%C4%99zie-Gita-Czym-jest-ga%C5%82%C4%85%C5%BA

Zmienianie gałęzi

Do przełączania się na gałąź służy komenda:

$ git checkout <NAZWA_GAŁĘZI>

Aby utworzyć nową gałąź i automatycznie się na nią przełączyć można użyć komendy:

$ git checkout -b [id_commitu]

Scalanie gałęzi

Merge czyli łączenie dwóch gałęzi.

Aby zmergować gałąź należy:

1. Przełączyć się na docelową gałąź:

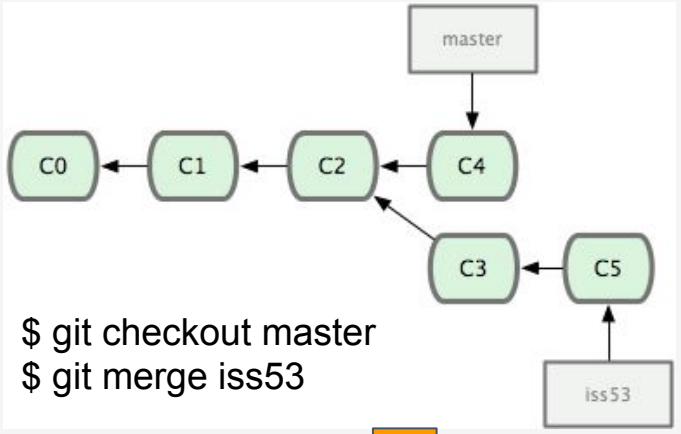
$ git checkout <DOCELOWA_GAŁĄŹ>

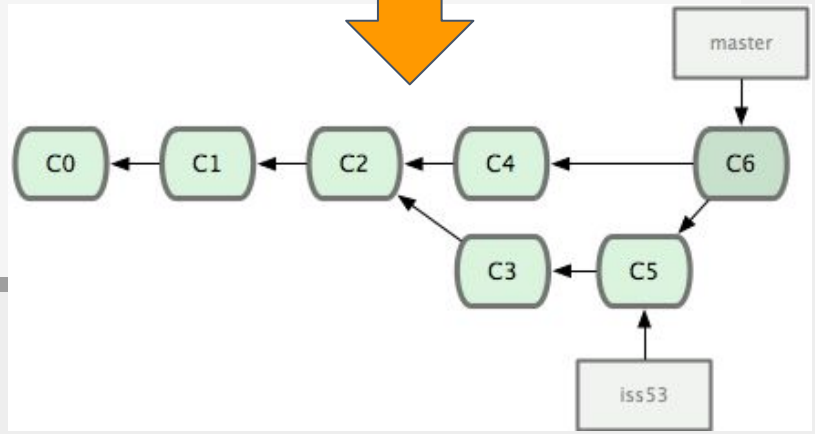
2. Wywołać komendę merge:

$ git merge <NAZWA_GAŁĘZI>

Warto zawsze sprawdzić aktualną gałąź

$ git branch





https://git-scm.com/book/pl/v1/Ga%C5%82%C4%99zie-Gita-Podstawy-rozga%C5%82%C4%99ziania-i-scalania

Usuwanie gałęzi

Aby usunąć zmergowaną gałąź:

$ git branch -d nazwa_galezi
branch_name nie może być aktywna.
Aby usunąć gałąź niezależnie od jej stanu:

$ git branch -D nazwa_galezi
Sprawdzenie w pełni zmergowanych gałęzi, może być przydatne przy kasowaniu. Aby wyświetlić gałęzie w pełni zmergowane (bezpiecznych do usunięcia):

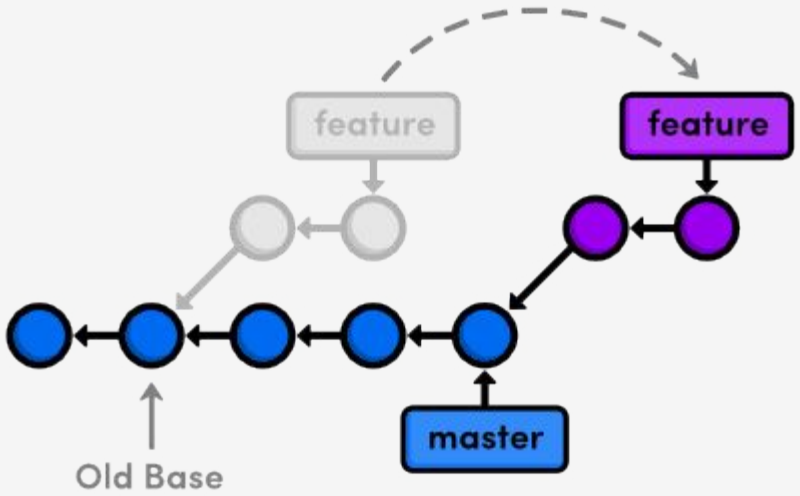
$ git branch --merged
Podobnie można wyświetlić te gałęzie, które mają nie zmergowane zmiany:
$ git branch --no-merged

Rebase

Rebase oznacza zmianę przodka naszej gałęzi

$ git rebase <zrodlowa_galaz>

Np: git checkout feature; git rebase master



Źródło: http://rypress.com/tutorials/git/rebasing

Push

Aby wgrać nasze zmiany na zdalne repozytorium należy użyć polecenia:

$ git push <zdalne_repo> <lokalna_galaz>:<zdalna_galaz>

Np:

$ git push origin szybkie-poprawki:poprawki-do-zadania-2

$ git push origin algorytm-przeszukiwania-drzewa-binarnego

Pull/Fetch

Do pobrania zmian ze zdalnego repozytorium służy komenda:

$ git fetch <zdalne_repo> <zdalna_galaz>

Do pobrania i automatycznego zmergowania zdalnej gałęzi do lokalnej służy komenda:

$ git pull <zdalne_repo> <zdalna_galaz> ==

$ git fetch <zdalne_repo> <zdalna_galaz>

$ git rebase origin <zdalna_galaz>

Poprawianie commitów

Zmiana ostatniego commitu, po co? a co co to komu?

1. Kiedy zapomnieliśmy dodać zmiany w jakimś pliku.
2. Szybkie poprawki.
3. Poprawienie opisu.
4. Kiedy z jakiegoś powodu trzeba go po prostu zmienić.

Poprawianie commitów 1/2

W gicie zmienianie ostatniego commitu to “amendowanie”.

1. Zmień, dodaj, usuń pliki jakie chcesz poprawić w ostatnim commicie
2. Dodaj wszystkie zmiany do śledzenia (git add/rm)
3. *$ git commit --amend*

Poprawianie commitów 2/2

“With great power comes great responsibility.” Ben “Uncle” Parker

NIE NALEŻY ZMIENIAĆ PUBLICZNYCH COMMITÓW!

Jeśli jednak już musisz to zrobić, to aby zaktualizować zdalną gałąź należy użyć opcji --force dla komendy push.

$ git push --force [-f]

Wycofywanie zmian 1/2

Aby wycofać plik z poczekalni (cofnąć polecenie git add/rm) należy użyć komendy:

$ git reset HEAD <plik>

Aby wycofać zmiany w zmodyfikowanym pliku (zmiany nie mogą być w poczekalni):

$ git checkout -- <plik>

Wycofywanie zmian 2/2

Jeżeli chcemy wycofać zmiany z commitu:

$ git revert <commit>
tworzy nowy commit, który wycofa nasze zmiany, które jednak ciągle zostaną w historii

$ git reset <commit>
przesuwa czubek gałęzi na podany commit, zmiany zostaną w folderze roboczym

$ git reset --hard <commit> **!!! POTENCJALNIE NIEBEZPIECZNE**
to samo co reset, ale czyści również folder roboczy. Usuwa zarówno commit jak i nasze zmiany!

.gitignore

Git umożliwia wyłączenie niektórych plików spod swoich operacji. Służą temu specjalne pliki „.gitignore”:

- Użytkownika C:/users/{myusername}/.gitignore lub “~” na Linuxie
- Repozytorium .gitignore w folderze naszego repo

Dodatkowe materiały

Do poćwiczenia:

https://try.github.io

http://learngitbranching.js.org/

Do poczytania:

https://git-scm.com/book/pl/v1

http://helion.pl/ksiazki/git-rozproszony-system-kontroli-wersji-wlodzimierz-gajda,gitroz.htm