

Git - część 1

System kontroli wersji



Cześć!

Daniel Kossakowski

DevOps Engineer @ CloudPanda.io

1.

Git? A co to?

Git? A co to?

- System kontroli wersji
- Pełna historia zmian
- Ułatwia rozwój i utrzymanie kodu
- Inne przykłady wersjonowania?
 - Microsoft Office
 - Google Docs

Git? Historia

- VCS:
 - Lokalne
 - Centralne
 - Rozproszone
- Rozwijany od 2005
- Pierwotnie posłużyć do prac na Linuxem

2.

Git? A komu to potrzebne?

A dlaczego?

Git? A komu to potrzebne?

- Praca w zespole
- Praca gdziekolwiek (lokalne operacje)

Git? A dlaczego?

- Szybkość
- Prosta konstrukcja
- Pełne rozproszenie
- Wsparcie dla nieliniowego rozwoju
- Wydajna obsługa dużych projektów

3.

Pierwsze repozytorium

Pierwsze repo Skąd wziąć Gita?

- Paczka w większości dystrybucji
- Ubuntu: Instalacja przez apt
- Windows: <https://desktop.github.com/>
- Wywołanie to po prostu: git

```
linux@iSA ~ $ sudo apt-get update
linux@iSA ~ $ sudo apt-get install git
linux@iSA ~ $ git --version
git version 2.17.1
```

Pierwsze repo Konfiguracja

- Każda zmiana posiada autora
- Jak git mnie rozpozna?
 - user.name - imię i nazwisko
 - user.email - adres e-mail (ten z Github.com)

```
linux@iSA ~ $ git config --global user.name "Jan Nowak"
```

```
linux@iSA ~ $ git config --global user.email "jannowak@example.com"
```

Pierwsze repo Inicjalizacja

- Repozytoria mogą być tworzone w dowolnych katalogach
- Można stworzyć repo w istniejącym folderze
- `git init`

Pierwsze repo Inicjalizacja

```
linux@iSA ~ $ mkdir /tmp/test
```

```
linux@iSA ~ $ cd /tmp/test
```

```
linux@iSA ~ $ git init
```

```
Initialized empty Git repository in /home/linux/.git/
```

```
linux@iSA ~ $ ls -la
```

```
drwxrwxr-x 3  linux linux 4096 paź 28 13:04 .  
drwxrwxrwt 21 root root 4096 paź 28 13:05 ..  
drwxrwxr-x 7  linux linux 4096 paź 28 13:04 .git
```

ĆWICZENIE!

- Utwórz folder o nazwie *repo1* w swoim katalogu domowym.
- Zainicjuj w nim nowe repozytorium.
- Sprawdź czy został utworzony katalog *.git*.

4.

Tworzenie zmian

Tworzenie zmian Status

- Sprawdzenie czy wprowadzono jakieś zmiany
- git status

```
linux@iSA ~ $ touch README.md
```

```
linux@iSA ~ $ git status
```

```
On branch master
```

```
No commits yet
```

```
Untracked files:
```

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

```
README.md
```

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


Tworzenie zmian Uwzględnianie plików

- Zawsze należy określić pliki, które zostaną uwzględnione
 - Nieuwzględnione nie zostaną dodane (Untracked files)
 - To nie jest jeszcze zapisanie zmiany!
 - Dobra praktyka: nie dodawać wszystkiego na raz
-
- `git add <nazwa pliku / folderu>`

Tworzenie zmian Uwzględnianie plików

```
linux@iSA ~ $ git add README.md
```

```
linux@iSA ~ $ git status
```

On branch master

No commits yet

Changes to be committed:

(use "git rm --cached <file>..." to unstage)

nowy plik: README.md

Tworzenie zmian **Zatwierdzenie**

- Zatwierdzenie zmiany to osobna komenda
 - Należy podać tytuł i opcjonalnie komentarz
 - Każda zmiana ma swój identyfikator (SHA)
-
- `git commit`
 - `git commit -m "Commit message"`
-
- `git commit -a` - zatwierdzenie wszystkiego
 - `git commit --amend` - nadpisanie poprzedniego commita

Tworzenie zmian Historia repo

- Lista commitów od początku
- Dodatkowe informacje: data, autor, identyfikator
- Polecenie: git log

Tworzenie zmian Historia repo

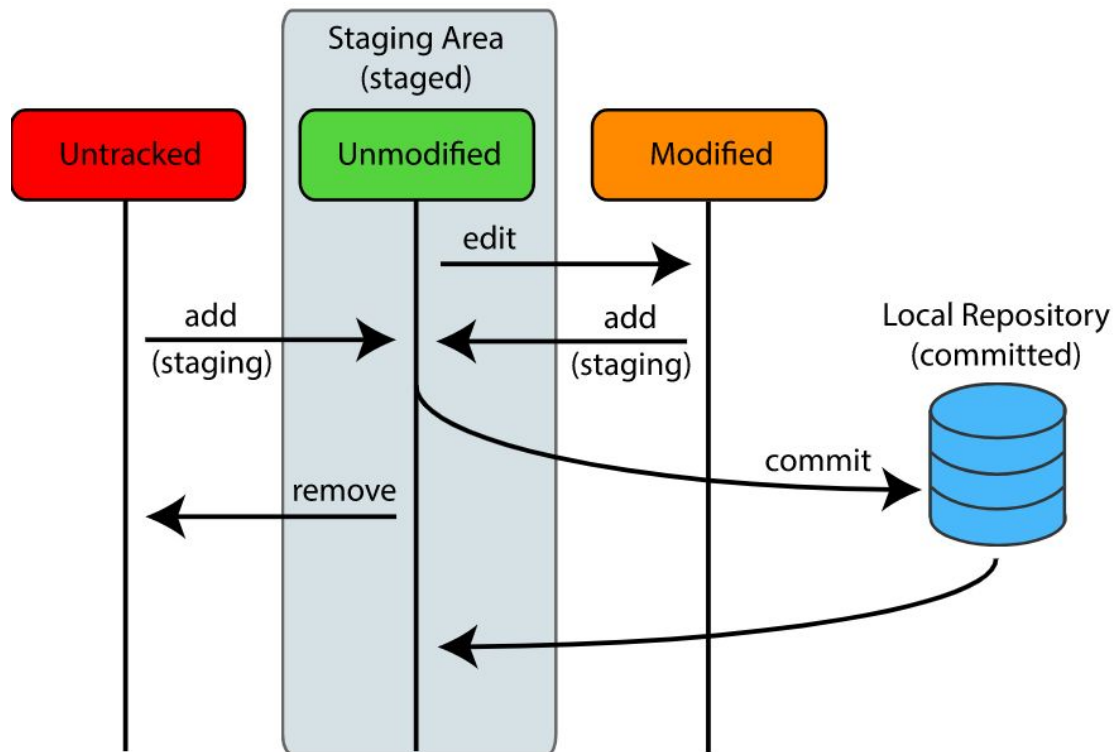
```
linux@iSA ~ $ git log
commit d5efd10c8be106c8f96de86060b882c056037376 (HEAD -> master, origin/master, origin/HEAD)
Author: Daniel Kossakowski <d.kossakowski@example.tld>
Date: Thu Sep 20 22:04:10 2018 +0200
```

Add missing files

```
commit 4c594318334fa6f26439aa4e169edb66f9356b7d
Merge: e243b6fd bccb25c4
Author: Jan Kowalski <j.kowalski@example.tld>
Date: Mon Sep 10 09:46:25 2018 +0200
```

Merge with develop

Tworzenie zmian Schemat



ĆWICZENIE!

- Utwórz pusty plik *test1* i *test2*.
- Zatwierdź te pliki w osobnych commitach.
- Sprawdź w logu czy się udało.

5. Odwracanie zmian

Odwracanie zmian **Reset**

- Odrzucenie zmian, które nie zostały zatwierdzone
- `git reset` - usuwa z poczekalni
- `git reset --hard` - przywraca stan do ostatniego commita

Odwracanie zmian **Revert**

- Wycofanie commita, który został już utworzony
- Powoduje utworzenie nowej zmiany 'revertującej'
- `git revert <SHA>`

ĆWICZENIE!

- W repozytorium z poprzedniego ćwiczenia zrób revert pierwszego commita.
- Sprawdź jak wygląda obecny status repozytorium.
- Sprawdź jak prezentuje się log.

6. Synchronizacja

Repozytoria współdzielone

- Wspólna praca zdalna nad jednym kodem
 - Kopia repozytorium u każdego
 - Lokalne i zdalne kopie
-
- `git clone <URL>`
 - `git clone <URL> <nazwa>`

```
linux@iSA ~ $ git clone https://github.com/infoshareacademy/jjdd5-materialy-git.git
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 3 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.
```

Pobieranie zmian

- `git pull` – pobranie zmian + merge
- `git fetch` – tylko pobranie zmian (bezpieczne)
- Wskazówka: warto zaczynać pracę od `git pull`

Przesyłanie zmian

- `git push` - przesłanie lokalnych zmian do repozytorium

```
linux@iSA ~ $ git push
Counting objects: 3, done.
Writing objects: 100% (3/3), 223 bytes | 223.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
remote:
remote: Create a pull request for 'master' on GitHub by visiting:
remote:   https://github.com/infoshareacademy/jjdd5-materialy-git/pull/new/master
remote:
To https://github.com/infoshareacademy/jjdd5-materialy-git.git
* [new branch] master -> master
```

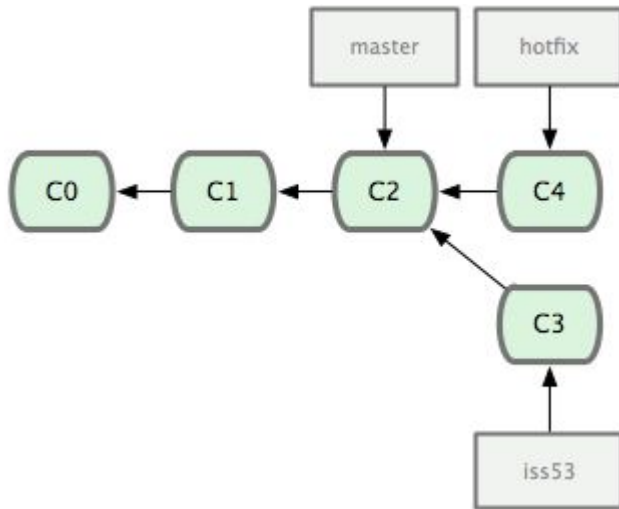
ĆWICZENIE!

- Pobierz repozytorium (*adres na Slacku*).
- Stwórz plik o nazwie *imie.nazwisko.txt*.
- Utwórz zmianę (commit)
- Prześlij zmianę do repozytorium.

7. Rozgałęzianie

Rozgałęzianie O co chodzi?

- Praca nad kilkoma zadaniami na raz
- Pozostawienie oryginalnego kodu bez zmian
- git pull / push może powodować konflikty



Rozgałęzianie Branch

- `git branch` - wyświetla gałęzie
 - `git branch -a` - wyświetla również zdalne gałęzie
 - `git branch -m` - zmiana nazwa gałęzi
-
- `git checkout <nazwa>` - przełącza na wskazaną gałąź / commit
 - `git checkout -b (...)` - tworzy nową gałąź i przełącza na nią

ĆWICZENIE!

- W repozytorium z poprzedniego ćwiczenia utwórz branch: *imie.nazwisko*
- Stwórz plik o nazwie *imie.nazwisko.txt.2*.
- Utwórz zmianę (commit)
- Prześlij zmianę do repozytorium na swój branch.

8. Konfiguracja

Konfiguracja Ignorowanie plików

- Git pozwala permanentnie ignorować pewne katalogi i pliki
- Repozytorium powinno trzymać kod, a nie konfiguracje
- Plik .gitignore – lista ignorowanych elementów
- Przykłady:
 - .env
 - .idea
 - *.bak

Konfiguracja Ignorowanie plików

- Puste katalogi są ignorowane domyślnie
- .gitkeep - rozwiązuje problem ignorowania

ĆWICZENIE!

- Przełącz się na gałąź master
- Utwórz feature branch change-1, utwórz pusty plik test-1 i go zatwierdź.
- Przełącz się na master.
- Utwórz feature branch change-2, utwórz pusty plik test-2 i go zatwierdź.
- Wróć do brancha master.
- Porównaj master z nowymi branchami (git diff).

9. Dobre praktyki

Dobre praktyki

Komentarze

Tytuły zmian powinny krótko opisywać i być zrozumiałe dla innych.

```
commit d8ba04bb9c0846d9527fa3c34338d0d901d37734
Merge: 1862f1e b100038
Author: Daniel Kossakowski <daniel.kossakowski@droptica.pl>
Date: Mon Oct 8 11:29:29 2018 +0000
```

Merged in develop (pull request #44)

Fix typo

```
commit b1000384bd122e749400d7244a6533561e7b90fe
Author: Daniel Kossakowski <daniel.kossakowski@droptica.pl>
Date: Mon Oct 8 13:28:44 2018 +0200
```

Fix typo

```
commit 0ccd3f96c5735d69a1c444fcabe5bf0ddf649c48
Author: Daniel Kossakowski <daniel.kossakowski@droptica.pl>
Date: Mon Oct 8 13:26:42 2018 +0200
```

backup: Add global DB_HOST

```
commit 437306e6415f37161d90b3527312d06300b19db2
Merge: 9e1566b d09bc3e
Author: Daniel Kossakowski <daniel.kossakowski@droptica.pl>
Date: Mon Oct 8 11:25:36 2018 +0000
```

Merged in develop (pull request #42)

Fix typo

```
commit d09bc3ee25f806d25f71863fc32a086333525231
Author: Daniel Kossakowski <daniel.kossakowski@droptica.pl>
Date: Mon Oct 8 13:24:59 2018 +0200
```

Fix typo

Dobre praktyki

Komentarze

Pierwsza linia: tytuł

Druga linia: opis

```
author      Changwei Ge <ge.changwei@h3c.com>      2018-11-02 15:48:15 -
committer   Linus Torvalds <torvalds@linux-foundation.org> 2018-11-03 10:09:37 -
commit      29aa30167a0a2e6045a0d6d2e89d8168132333d5 (patch)
tree        42c5c20786993ef2a0f4dd7e9411cc0bd80a78f3
parent      9e985787750db8aae87f02b67e908f28ac4d6b83 (diff)
download    linux-29aa30167a0a2e6045a0d6d2e89d8168132333d5.tar.gz
```

ocfs2: fix a misuse a of brelse after failing ocfs2_check_dir_en

Somehow, file system metadata was corrupted, which causes ocfs2_check_dir_entry() to fail in function ocfs2_dir_foreach_blk_el().

According to the original design intention, if above happens we should skip the problematic block and continue to retrieve dir entry. But there is obviouse misuse of brelse around related code.

After failure of ocfs2_check_dir_entry(), current code just moves to next position and uses the problematic buffer head again and again during which the problematic buffer head is released for multiple times. I suppose, this a serious issue which is long-lived in ocfs2. This may cause other file systems which is also used in a the same host insane.

So we should also consider about bakcporting this patch into linux -stable.

Dobre praktyki Nazewnictwo branchy

- Tworzymy branch per zmianę (feature branch)
- Rodzaje gałęzi:
 - Master
 - Develop
 - Hotfix



Koniec!

Dziękuję za uwagę

Pytania?