10 Sekretnych Komend Gita, O Których Nie Masz Pojęcia

(UWAGA: Twój Git-Skill Wzrośnie W Kilka Minut!)

Maciej Aniserowicz

Cześć!

Jestem Maciej Aniserowicz z devstyle.pl ,
a Ty jesteś... we właściwym miejscu,
by nauczyć się Gita! 👊

Przygotowałem dla Ciebie moją subiektywną listę
👉 *10 nieoczywistych komend*
tego wspaniałego narzędzia.

Chcę pokazać Ci, że **Git to nie jest głupi tool**, służący tylko do generowania ZIPów.

Smacznego!

01 pickaxe

Co robi:

- wyszukuje commity zawierające (**w treści zmian**) dany tekst

Przyda Ci się, jeśli:

- szukasz przyczyny modyfikacji w konkretnej linii, a **git blame zawodzi** (*zdarza się †*)
- nie możesz znaleźć przyczyny modyfikacji z powodu **zmiany nazwy lub skasowania pliku** (*też się zdarza*)
- chcesz znaleźć wszystkie commity zawierające dany tekst (np. **wrażliwe dane**)

01 pickaxe

Sposób użycia

1. git log -S <text>
2. dodaj przełącznik -p , by od razu zobaczyć zawartość commita, a nie tylko metadane

Przełącznik -S to właśnie tytułowa funkcja "pickaxe"

02 push --force-with-lease

Co robi:

- nadpisuje zmiany na serwerze, ale...
- ...upewniając się, że masz ściągniętą najnowszą wersję gałęzi
- (czyli: **wiesz, co nadpisujesz** 😇)

Przyda Ci się, jeśli:

- **absolutnie musisz nadpisać historię** zdalnego repozytorium (--force)
- zawsze zamiast tego użyj --force-with-lease

02 push --force-with-lease

Sposób użycia

1. git push --force-with-lease

UWAGA: Nigdy nie nadpisuj publicznej historii, chyba że absolutnie musisz!

03 commit --fixup

Co robi:

- **automatycznie scala** (squash) aktualny commit z innym wybranym commitem podczas rebase
- oznacza aktualny commit jako "*fixup*" (do poczytania przy okazji *interactive rebase*)

Przyda Ci się, jeśli:

- lubisz mieć **porządek w historii** 👏
- pilnujesz, by commity były zgodne z **Single Responsibility Principle** 👏👏
- chcesz **dokleić** zmiany do już istniejącego commita

03 commit --fixup

Sposób użycia

1. włącz w konfiguracji: git config rebase.autosquash true
2. wykonaj: git commit --fixup=<SHA1>
3. przejdź w tryb interactive rebase: git rebase -i <SHA1>^ i zobacz, że commit został przesunięty w historii
4. zaakceptuj proponowane operacje, by aktualny commit został scalony z commitem <SHA1>

Gdzie SHA1 to identyfikator commita, do którego *chcesz się dokleić*.

04 --assume-unchanged

Co robi:

- ignoruje zmiany w wybranych plikach **dodanych już do repozytorium**

Przyda Ci się, jeśli:

- często modyfikujesz jakiś plik (na przykład uzupełniasz **placeholdery w konfiguracji**), ale nie chcesz zapisywać tych zmian do repozytorium
- chcesz tymczasowo oznaczyć **failing test jako ignorowany**, ale nie wysłać tej informacji dalej ⚡
- **żałujesz, że .gitignore nie potrafi** ignorować plików dodanych do repozytorium 😭

04 --assume-unchanged

Sposób użycia

1. włączenie: git update-index --assume-unchanged <path>
2. wyłączenie: git update-index --no-assume-unchanged <path>
3. wylistowanie oznaczonych w ten sposób plików: git ls-files -v | grep '^[[:lower:]]'
4. przydany alias: ignored = !git ls-files -v | grep "^[[:lower:]]"

05 sparse checkout

Co robi:

- pozwala zrobić "checkout" jedynie wybranych podkatalogów repozytorium

Przyda Ci się, jeśli:

- pracujesz na **dużym repozytorium Gita**, a potrzebujesz tylko jego części do pracy
- używasz Gita do pracy z **SVN lub TFS** □
- chcesz **zmniejszyć wielkość** working copy
- chcesz **zwiększyć wydajność** Gita na dużym repozytorium

05 sparse checkout

Sposób użycia

1. włącz w konfiguracji: git config core.sparsecheckout true
2. utwórz plik: .git/info/sparse-checkout
3. uzupełnij plik wybranymi katalogami, jeden wpis per linia
   3.1. np. Dev/Main
   3.2. np. Config/Scripts
   3.3. można stosować patterns i negację (wykrzyknik z przodu) ⚠
4. zaktualizuj working copy: git read-tree -m -u HEAD

06 aliasy: wykrzyknik i &&

Co robi:

- pozwala wykonać **wiele komend w jednym aliasie** 😀

Przyda Ci się, jeśli:

- chcesz utworzyć alias dla potencjalnie złożonej procedury 🤔

06 aliasy: wykrzyknik i &&

Sposób użycia

1. w pliku konfiguracyjnym zdefiniuj alias, stawiając przed nim wykrzyknik
2. połącz komendy podwójnym znakiem &

Przykład

[alias]

stl = !git status && git log -n1

(git stl => wykona status i pokaże log ostatniego commita)

07 RERERE

Co robi:

- zapamiętuje **rozwiązania konfliktów** i używa ich w przyszłości
- "**REuse REcorded REsolution** of conflicted merges” 📚

Przyda Ci się, jeśli:

- nie chcesz wielokrotnie borykać się z **tymi samymi konfliktami** podczas merge (a kto chce? 😎)

07 RERERE

Sposób użycia

1. włącz w konfiguracji: git config rerere.enabled 1
2. enjoy  (*to nie jest komenda Gita*)

08 rebase -i --root

Co robi:

- pozwala zrobić **rebase pierwszego commita** w historii

Przyda Ci się, jeśli:

- wiesz, że czasem trzeba przepisać **całą historię projektu od nowa**, a jednocześnie ...
- ... nie chcesz rozpoczynać każdego projektu od pustego commita ...
- ..."*na wszelki wypadek*", gdyby kiedyś trzeba było zrobić pełen rebase od samego początku

08 rebase -i --root

Sposób użycia

1. git rebase -i --root

09 bisect

Co robi:

- bardzo sprawnie **wyszukuje konkretny pojedynczy commit** powodujący zmianę w zachowaniu systemu
- używa algorytm **binary search** 🎓
- między innymi dla tej komendy warto dbać o **małe, spójne commity**! !!

Przyda Ci się, jeśli:

- chcesz znaleźć **przyczynę powstania błędu !?** - konkretny commit powodujący zmianę zachowania
- (uwaga: działa najlepiej przy wielu małych commitach)

09 bisect

Sposób użycia (1)

1. (założenie: jesteś na master i coś nie działa) 😠
2. znajdź dowolny commit, który działa tak jak trzeba (na przykład sprzed miesiąca): git checkout <SHA1>
3. zweryfikuj, czy na pewno działa 👍
4. rozpocznij proces bisect: git bisect start
5. oznacz aktualny commit jako poprawny: git bisect good 👌
6. oznacz najnowszy commit jako niepoprawny: git bisect bad master 👎

09 bisect

Sposób użycia (2)

6. dla każdego podanego przez Git commita:
   7.1 sprawdź, czy działa
   7.2 jeśli tak, napisz: git bisect good
   7.3 jeśli nie, napisz: git bisect bad
7. szybko znajdziesz commit odpowiedzialny za wprowadzenie zmiany
8. zakończ proces bisect: git bisect reset 👏

10 filter-branch

Co robi:

- **dowolnie modyfikuje** historię gałęzi 

Przyda Ci się, jeśli:

- chcesz usunąć z **całego** repozytorium **plik z hasłem** lub **duży plik**
- **--amend** nie wystarcza
- **rebase** nie wystarcza
- chcesz **masowo** <zrobić cokolwiek> z wieloma commitami

⚠ **Zalecenie**: przygotuj **nową gałąź** do eksperymentów: git checkout -b experiments

10 filter-branch

Przykład 1

Dodaj [prefix] do wszystkich commit messages

```
git filter-branch --msg-filter \
'read m;
echo "[prefix] $m" ,
```

(sprawdź: git log)

10 filter-branch

Przykład 2

Przenieś wszystkie pliki z wszystkich commitów do podkatalogu "Dev"

```
git filter-branch --tree-filter \
'mkdir Dev
ls -A | grep -v Dev | while read filename
do
mv $filename Dev
done'
```

10 filter-branch

Przykład 3

Zmień datę wszystkich commitów (na wszystkich gałęziach) na dzień "narodzin" Gita (7 kwietnia 2005)

```
git filter-branch --env-filter '
export GIT_AUTHOR_DATE="2005-04-07 00:00:00"
export GIT_COMMITTER_DATE="2005-04-07 00:00:00"
' -- --all
```

KONIEC

Jeśli choć jedno z tych poleceń było dla Ciebie nowością, to już **możesz wycisnąć z Gita więcej**, niż jeszcze kilka minut temu!

Mission accomplished! 👏

Masz uwagi, sugestie? Pamiętaj, że zawsze jestem dla Ciebie dostępny pod adresem maciej@devstyle.pl.

Miłego dnia! 

Let the GitForce be with you!

BONUS pager less -r

Co robi:

- włącza "word wrap" (zawijanie wierszy) **podczas stronicowania tekstu w Gicie**

Przyda Ci się, jeśli:

- chcesz widzieć "zawijane" długie linie (word wrap) przy komendach takich jak **log czy diff** zamiast uciętych linii, wymagających scrollowania na boki

BONUS pager less -r

Sposób użycia

1. ustaw pager dla jednej komendy:
   git_pager='less -r' git diff
2. LUB ustaw pager w konfiguracji
   git config core.pager 'less -r'

Maciej Aniserowicz

Programista-pasjonat.
Animator polskiej społeczności IT.
Twórca bloga i VLOGa devstyle, podcasta DevTalk.
Autor bestsellerowej książki "Zawód: Programista"
oraz kursu Gita: KursGita.pl.

E-mail
maciej@devstyle.pl

Twitter
@maniserowicz

Instagram
@maciej.aniserowicz

www
devstyle.pl