

Helion 



Git

Włodzimierz Gajda

Rozproszony system
kontroli wersji

Wszelkie prawa zastrzeżone. Nieautoryzowane rozpowszechnianie całości lub fragmentu niniejszej publikacji w jakiegokolwiek postaci jest zabronione. Wykonywanie kopii metodą kserograficzną, fotograficzną, a także kopiowanie książki na nośniku filmowym, magnetycznym lub innym powoduje naruszenie praw autorskich niniejszej publikacji.

Wszystkie znaki występujące w tekście są zastrzeżonymi znakami firmowymi bądź towarowymi ich właścicieli.

Autor oraz wydawca dołożyli wszelkich starań, by zawarte w tej książce informacje były kompletne i rzetelne. Nie biorą jednak żadnej odpowiedzialności ani za ich wykorzystanie, ani za związane z tym ewentualne naruszenie praw patentowych lub autorskich. Autor oraz wydawca nie ponoszą również żadnej odpowiedzialności za ewentualne szkody wynikłe z wykorzystania informacji zawartych w książce.

Redaktor prowadzący: Ewelina Burska

Projekt okładki: Jan Paluch

Materiały graficzne w książce i na okładce zostały wykorzystane za zgodą Shutterstock.com

Helion S.A.

ul. Kościuszki 1c, 44-100 Gliwice

tel. 32 231 22 19, 32 230 98 63

e-mail: helion@helion.pl

WWW: <https://helion.pl> (księgarnia internetowa, katalog książek)

Drogi Czytelniku!

Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres

<https://helion.pl/user/opinie/gitrov>

Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

ISBN: 978-83-283-9520-6

Copyright © Helion S.A. 2013, 2022

Printed in Poland.

- [Kup książkę](#)
- [Poleć książkę](#)
- [Oceń książkę](#)

- [Księgarnia internetowa](#)
- [Lubię to! » Nasza społeczność](#)

Spis treści

Podziękowania	9
Część I Repozytoria o liniowej historii	11
Rozdział 1. Wprowadzenie	13
Git	13
Jak przebiega praca nad projektem stosującym Git?	14
Hosting projektów Git	19
Czego się nauczysz z tego podręcznika?	20
Dokumentacja	20
Rozdział 2. Instalacja programu Git	23
Konsola Gita w systemie Windows	25
Ułatwienia uruchamiania konsoli w systemie Windows	26
Podstawowa konfiguracja klienta Git	27
Edytor	28
Rozdział 3. Tworzenie repozytoriów	29
Inicjalizacja nowego repozytorium	29
Klonowanie repozytoriów	30
Badanie historii projektu	33
Wizualizacja historii projektu	36
Rozdział 4. Obszar roboczy	39
Przywracanie stanu projektu, który zawiera nowe pliki	41
Rozdział 5. Tworzenie rewizji i przywracanie stanu plików	43
Tworzenie rewizji	43
Przywracanie stanu plików do wybranej rewizji	45
Przenoszenie repozytorium	48
Rezygnacja z repozytorium	49
Rozdział 6. Stany plików	51
Uproszczony model pracy: przestrzeń robocza i repozytorium	51
Indeksowanie	52
Diagram stanów	53

Obszar roboczy, indeks i repozytorium	56
Modyfikowanie stanu plików repozytorium	57
Stan repozytorium	61
Uproszczony model pracy raz jeszcze	62
Oznaczenia stanów pliku	68
Stany dwuliterowe (mieszane)	69
Repozytoria zwykłe i surowe	72
Składnia poleceń Gita	73
Rozdział 7. Ignorowanie plików	75
Uzupełnienie diagramu stanów	78
Rozdział 8. Znaczniki	83
Znaczniki lekkie i oznaczone	83
Tworzenie znaczników opisanych	84
Tworzenie znaczników lekkich	84
Usuwanie znaczników	85
Sprawdzanie dostępnych znaczników	85
Szczegółowe dane znacznika	85
Użycie znaczników	86
Generowanie skompresowanych plików odpowiadających konkretnej wersji projektu	89
Rozdział 9. Identyfikowanie rewizji	91
Pełne skróty SHA-1	91
Skrócona postać SHA-1	92
Znaczniki	92
Nazwa symboliczna HEAD	93
Rewizja domyślna	93
Repozytoria o historii nieliniowej	94
Dziennik reflog	100
Polecenia rev-parse oraz rev-list	101
Znaki specjalne wiersza poleceń Windows	102
Rozdział 10. Skróty komend	107
Komendy ułatwiające zapisywanie stanu projektu	108
Komendy ułatwiające wykonywanie ćwiczeń	110
Rozdział 11. Modyfikowanie historii projektu	115
Usuwanie ostatnich rewizji	116
Modyfikowanie ostatniej rewizji	117
Łączenie rewizji	117
Usuwanie zmian wprowadzonych przez rewizję	120
Odzyskiwanie poszczególnych plików z dowolnej rewizji	125
Rozdział 12. Podsumowanie części I	127
Co powinieneś umieć po lekturze pierwszej części?	130
Lista poznanych poleceń	130

Część II Repozytoria z rozgałęzieniami	139
Rozdział 13. Tworzenie i usuwanie gałęzi	141
Gałęzie to wskaźniki rewizji!	141
Gałąź master	141
Tworzenie gałęzi	143
Dodawanie rewizji w bieżącej gałęzi	143
Tworzenie gałęzi wskazujących dowolną rewizję	144
Przełączanie gałęzi	145
Tworzenie i przełączanie gałęzi	147
Stan detached HEAD	148
Relacja zawierania gałęzi	150
Usuwanie gałęzi	153
Zmiana nazwy gałęzi	155
Gałęzie jako identyfikatory rewizji	156
Uwagi o usuwaniu ostatnich rewizji	157
Sprawdzanie różnic pomiędzy gałęziami	157
Gałęzie i dziennik reflog	161
Zgubione rewizje	163
Rozdział 14. Łączenie gałęzi: operacja merge	167
Przewijanie do przodu	168
Przewijanie do przodu dla wielu gałęzi	169
Łączenie gałęzi rozłącznych	170
Łączenie kilku rozłącznych gałęzi	171
Wycofywanie operacji git merge	173
Rozdział 15. Łączenie gałęzi: operacja rebase	175
Podobieństwa i różnice pomiędzy poleceniami merge i rebase	176
Wycofywanie operacji git rebase	178
Rozdział 16. Podsumowanie części II	181
Co powinieneś umieć po lekturze drugiej części?	181
Lista poznanych poleceń	182
Część III Gałęzie zdalne	185
Rozdział 17. Definiowanie powiązania między repozytorium lokalnym a zdalnym	187
Klonowanie raz jeszcze	187
Klonowanie repozytorium z dysku	191
Definiowanie repozytoriów zdalnych	192
Definiowanie powiązania między gałęzią lokalną a gałęzią śledzoną	193
Listowanie gałęzi	194
Rozdział 18. Podstawy synchronizacji repozytoriów	195
Pobieranie gałęzi z repozytorium zdalnego do repozytorium lokalnego	195
Uaktualnianie sklonowanych repozytoriów	197
Repozytoria surowe	198
Przesyłanie gałęzi do repozytorium zdalnego	199

Wysyłanie dowolnej gałęzi	206
Przełączanie na gałąź zdalną	208
Przesyłanie gałęzi ze zmianą nazwy	208
Usuwanie gałęzi zdalnych	209
Zabezpieczanie przed utratą rewizji	209
Polecenie backup	210
Przesyłanie gałęzi do repozytorium zwykłego	210
Rozdział 19. Praktyczne wykorzystanie Gita — scenariusz pierwszy	215
Inicjalizacja projektu	216
Dołączanie do projektu	216
Wprowadzanie zmian w projekcie	217
Wykorzystywanie kilku gałęzi	218
Rozdział 20. Łączenie oddzielnych repozytoriów	219
Graf niespójny	223
Rozdział 21. Podsumowanie części III	225
Co powinieneś umieć po lekturze trzeciej części?	226
Lista poznanych poleceń	226
Część IV Treść pliku	231
Rozdział 22. Konflikty	233
Konflikt tekstowy: wynik operacji git merge	233
Konflikt tekstowy: wynik operacji git rebase	236
Dublowanie konfliktów przez operacje merge i rebase	238
Konflikty binarne	238
Konflikt binarny: wynik operacji git merge	239
Konflikt binarny: wynik operacji git rebase	240
Przywracanie plików do postaci z łączonych gałęzi	242
Polecenia checkout i show	242
Rozdział 23. Badanie różnic	245
Szukanie zmienionych wyrazów	253
Szukanie zmienionych plików	254
Wyszukiwanie rewizji, w których podany plik został zmieniony	255
Rozdział 24. Pliki tekstowe i binarne	257
Odróżnianie plików binarnych od tekstowych	257
Atrybut diff — konflikty tekstowe i binarne	258
Konwersja znaków końca wiersza	259
Projekty wieloplatformowe	260
Ustalenie konwersji znaków końca wiersza dla konkretnych plików	261
Rozdział 25. Podsumowanie części IV	263
Co powinieneś umieć po lekturze czwartej części?	263
Lista poznanych poleceń	264

Część V Praca w sieci	267
Rozdział 26. Serwisy github.com i bitbucket.org	269
Rozdział 27. Klucze SSH	277
Instalacja oprogramowania SSH w systemie Windows	277
Konfiguracja klucza SSH na serwerze github.com	279
Konfiguracja klucza SSH na serwerze bitbucket.org	280
Repozytorium zdalne na serwerze SSH	280
Rozdział 28. Tworzenie i usuwanie repozytoriów w serwisach github.com i bitbucket.org	283
Inicjalizowanie nowego repozytorium: serwis github.com	283
Import istniejącego kodu: serwis github.com	286
Inicjalizowanie nowego repozytorium: serwis bitbucket.org	287
Rozdział 29. Praktyczne wykorzystanie Gita — scenariusz drugi	291
Scenariusz pierwszy realizowany w serwisach github.com i bitbucket.org	292
Rozdział 30. Praca grupowa w serwisach github.com oraz bitbucket.org	293
Praca oparta na żądaniach aktualizacji	294
Praca grupowa wykorzystująca żądania aktualizacji (bez gałęzi) w pigułce	301
Żądania aktualizacji i gałęzie	303
Opisy i dyskusje	313
Rozdział 31. Zintegrowany system śledzenia błędów	315
Rozdział 32. Podsumowanie części V	319
Repozytoria do ćwiczenia znajomości Gita	319
Dodatki	321
Dodatek A Literatura	321
Dodatek B Słownik terminów angielskich	323
Skorowidz	325

Rozdział 17.

Definiowanie powiązania między repozytorium lokalnym a zdalnym

Przejdźmy do definiowania powiązań pomiędzy repozytoriami. Repozytorium, w którym poleceniami `git add` oraz `git commit` będziemy wykonywali rewizje, nazwiemy **repozytorium lokalnym**. Repozytoria, które posłużą do synchronizacji rewizji, nazwiemy **repozytoriami zdalnymi**.

Repozytoria lokalne będą repozytoriami zwykłymi, a repozytoria zdalne — repozytoriami surowymi.

Klonowanie raz jeszcze

Poznana w rozdziale 3. operacja klonowania:

```
git clone adres [folder]
```

dotyczy dwóch repozytoriów. Na lokalnym dysku tworzymy repozytorium, które będzie kopią repozytorium zdalnego. Klonowanie jest więc najprostszym przykładem tworzenia powiązania pomiędzy dwoma repozytoriami.

Po wykonaniu operacji klonowania repozytorium utworzone na dysku nazwiemy **repozytorium lokalnym**, a repozytorium, którego adres pojawił się w poleceniu `git clone` — **repozytorium zdalnym**.



Wskazówka

Po wykonaniu operacji:

```
git clone git://github.com/symfony/symfony.git .
```

repozytorium:

```
git://github.com/symfony/symfony.git
```

jest **repozytorium zdalnym**. Repozytorium utworzone na dysku to **repozytorium lokalne**.

Podczas klonowania wykonywane są następujące czynności:

1. Proces rozpoczyna się od inicjalizacji nowego pustego repozytorium lokalnego.
2. W repozytorium lokalnym dodawany jest adres repozytorium zdalnego. Adres ten jest automatycznie oznaczany nazwą `origin`.
3. Z repozytorium zdalnego kopiowane są rewizje ze zdalnej gałęzi `master` do lokalnej gałęzi `master`.
4. W repozytorium lokalnym w folderze `.git/refs/remotes/origin` tworzony jest plik `HEAD` zawierający nazwę symboliczną domyślnej gałęzi repozytorium zdalnego.
5. Następnie definiowane jest powiązanie lokalnej gałęzi `master` ze zdalną gałęzią `master`. Gałąź lokalna będzie **śledziła** (ang. *track*) gałąź zdalną.
6. Na zakończenie stan plików w obszarze roboczym repozytorium lokalnego jest przywracany do postaci z gałęzi `master`.

Wszystkie powyższe operacje możemy wykonać ręcznie.

Procedura ręcznego klonowania

1. Polecenie:

```
git init
```

tworzy nowe puste repozytorium `git`.

2. Polecenie:

```
git remote add origin adres
```

dodaje w konfiguracji adres repozytorium zdalnego.

3. Polecenie:

```
git fetch --no-tags origin master:refs/remotes/origin/master
```

kopiuje z repozytorium zdalnego do repozytorium lokalnego wszystkie rewizje zawarte w gałęzi `master`. Ponadto w repozytorium lokalnym w folderze `.git/refs/remotes/origin` tworzona jest nazwa symboliczna dla zdalnej gałęzi `master`. Parametr `--no-tags` powoduje, że znaczniki z repozytorium zdalnego nie będą kopiowane.

4. Polecenie:

```
git branch --set-upstream master origin/master
```

tworzy powiązanie pomiędzy lokalną gałęzią `master` a zdalną gałęzią `master`.

5. Na zakończenie polecenie:

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

przywraca stan plików w obszarze roboczym.



Wskazówka

Repozytorium wykonane opisaną powyżej procedurą różni się od repozytorium klonowanego tylko tym, że w repozytorium klonowanym w pliku `refs/remotes/origin/HEAD` adres gałęzi zdalnej `master` jest zapisany w postaci symbolicznej:

```
ref: refs/remotes/origin/master
```

Ćwiczenie 17.1

Sklonuj repozytorium jQuery:

```
git://github.com/jquery/jquery.git
```

po czym sprawdź konfigurację repozytorium lokalnego.

Po wykonaniu polecenia:

```
git clone git://github.com/jquery/jquery.git .
```

w repozytorium lokalnym w pliku `.git/config` znajdziemy wpisy ustalające adres `origin` oraz powiązanie lokalnej gałęzi `master` ze zdalną gałęzią `master`:

```
[remote "origin"]
  fetch = +refs/heads/*:refs/remotes/origin/*
  url = git://github.com/jquery/jquery.git
[branch "master"]
  remote = origin
  merge = refs/heads/master
```

W pliku `.git/HEAD` znajdziemy odwołanie symboliczne:

```
ref: refs/heads/master
```

W pliku `.git/refs/heads/master` znajdziemy skrót SHA-1 rewizji, a w pliku `.git/refs/remotes/origin/HEAD` — odwołanie symboliczne:

```
ref: refs/remotes/origin/master
```

Ćwiczenie 17.2

Wykorzystując polecenia:

```
git init
git remote add origin git://github.com/jquery/jquery.git
git fetch --no-tags origin master:refs/remotes/origin/master
```

```
git branch --set-upstream master origin/master
git reset --hard HEAD
```

sklonuj repozytorium jQuery:

```
git://github.com/jquery/jquery.git
```

Przed i po wykonaniu każdego kroku sprawdź zawartość następujących plików i folderów konfiguracyjnych:

```
.git/config
.git/HEAD
.git/refs/heads/master
.git/refs/remotes/origin
```

ROZWIĄZANIE

Krok 1.

Utwórz nowy folder i wydaj w nim polecenie:

```
git init
```

W tym momencie plik `.git/config` nie zawiera żadnych informacji o repozytoriach zdalnych. W pliku `.git/HEAD` obecne jest odwołanie symboliczne:

```
ref: refs/heads/master
```

W folderze `.git/refs` nie występują plik `.git/refs/heads/master` i folder `.git/refs/remotes`.

Krok 2.

Po wydaniu polecenia:

```
git remote add origin git://github.com/jquery/jquery.git
```

w pliku `.git/config` znajdziemy wpis:

```
[remote "origin"]
  url = git://github.com/jquery/jquery.git
  fetch = +refs/heads/*:refs/remotes/origin/*
```

który ustala adres repozytorium określanego nazwą symboliczną `origin`.

Zawartość folderu `.git/refs/` i pliku `.git/HEAD` nie uległa zmianie.

Krok 3.

Wydaj polecenie:

```
git fetch --no-tags origin master:refs/remotes/origin/master
```

Spowoduje ono pobranie z repozytorium zdalnego `origin` wszystkich rewizji zawartych w gałęzi `master`.

Ponadto w folderze `.git/refs/remotes/origin/master` utworzony zostanie plik zawierający skrót SHA-1 ostatniej rewizji w gałęzi `master` repozytorium zdalnego `origin`.

Krok 4.

Wydaj polecenie:

```
git branch --set-upstream master origin/master
```

W ten sposób zdefiniowane zostanie powiązanie pomiędzy lokalną gałęzią `master` a zdalną gałęzią `master`. Powiązanie to jest zapisywane w pliku `.git/config` w postaci wpisu:

```
[branch "master"]
  remote = origin
  merge = refs/heads/master
```

Krok 5.

Ostatnie z poleceń:

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

przywraca stan plików obszaru roboczego do postaci z ostatniej rewizji zawartej w lokalnej gałęzi `master`.

Klonowanie repozytorium z dysku

Operację klonowania repozytorium możemy wykonać lokalnie, bez żadnej komunikacji sieciowej. Adresem repozytorium zdalnego może być ścieżka prowadząca do repozytorium. Polecenie:

```
git clone C:\my\repos\example .
```

klonuje repozytorium z folderu `C:\my\repos\example` do folderu bieżącego.

Ścieżkę prowadzącą do repozytorium zdalnego możemy także przekazać jako parametr polecenia `git remote`, np.:

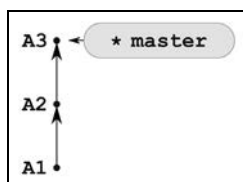
```
git remote add origin C:\my\repos\example
```

Dzięki takiemu rozwiązaniu ćwiczenia dotyczące synchronizacji repozytoriów będziemy mogli wykonywać w pełni lokalnie.

Ćwiczenie 17.3

W folderze `cw-17-03/` wykonaj repozytorium przedstawione na rysunku 17.1.

Rysunek 17.1.
Repozytorium
z ćwiczenia 17.3



Ćwiczenie 17.4

Repozytorium z ćwiczenia 17.3 sklonuj do folderu *cw-17-04/*.

ROZWIĄZANIE

Przyjmijmy, że foldery:

```
cw-17-03/  
cw-17-04/
```

znajdują się w tym samym folderze.

W wierszu poleceń przejdź do folderu *cw-17-04/* i wydaj komendę:

```
git clone ../cw-17-03 .
```

Alternatywnie klonowanie możesz wykonać, wydając polecenie:

```
git clone cw-17-03 cw-17-04
```

w folderze zawierającym foldery *cw-17-03/* oraz *cw-17-04/*.

Po tej operacji w folderze *cw-17-04/* znajdziemy kopię repozytorium z ćwiczenia 17.3. Ponadto w pliku *.git/config* znajdziemy wpisy:

```
[remote "origin"]  
  fetch = +refs/heads/*:refs/remotes/origin/*  
  url = C:/git/cw-17-04/ ../cw-17-03  
[branch "master"]  
  remote = origin  
  merge = refs/heads/master
```

Definiowanie repozytoriów zdalnych

Do ustalenia adresu repozytorium zdalnego służy komenda:

```
git remote add nazwa adres
```

Parametrem *nazwa* określamy sposób odwoływania się do definiowanego repozytorium zdalnego, a parametr *adres* określa jego adres. Polecenie `git remote add` zapisuje informacje o repozytorium zdalnym w pliku *.git/config*. Przykładowy wpis przyjmuje postać przedstawioną na listingu 17.1.

Listing 17.1. Fragment pliku *.git/config* zawierający informacje o repozytorium zdalnym *nazwa*

```
[remote "nazwa"]  
  url = adres  
  fetch = +refs/heads/*:refs/remotes/nazwa/*
```

Jeśli repozytorium znajduje się na dysku w folderze `C:\repos\zdalne` i zechcemy mu nadać nazwę zdalne, polecenie `git remote` przyjmie wówczas postać:

```
git remote add zdalne C:\repos\zdalne
```

W kolejnych rozdziałach repozytoria zdalne będą pochodziły z serwerów *github.com* oraz *bitbucket.org*. Polecenie `git remote` przyjmie wówczas postać:

```
git remote add my git@github.com:gajdaw/symfony.git  
git remote add gajdaw git@bitbucket.org:gajdaw/symfony.git
```

Do wyświetlenia listy repozytoriów zdalnych służy polecenie:

```
git remote -v
```

Adres repozytorium zdalnego możemy usunąć poleceniem:

```
git remote rm nazwa
```

Definiowanie powiązania między gałęzią lokalną a gałęzią śledzoną

Dla każdej gałęzi zawartej w repozytorium lokalnym możemy ustalić odpowiadającą jej **gałąź śledzoną** (ang. *tracking branch*). Dzięki temu polecenia synchronizacji, np.:

```
git pull  
git push  
git fetch
```

mogą być wywoływane bez parametrów. W takiej sytuacji synchronizacja będzie dotyczyła bieżącej gałęzi oraz odpowiadającej jej gałęzi śledzonej.

Nazwy gałęzi śledzonych poznamy, wydając polecenie:

```
git config --list
```

Wydruk będzie zawierał informacje postaci:

```
branch.master.remote=origin  
branch.master.merge=refs/heads/master
```

Ogólnie rzecz biorąc, dla gałęzi lokalnej o nazwie `X` wpisy ustalające gałąź śledzoną będą następujące:

```
branch.X.remote=...  
branch.X.merge=...
```



Wskazówka

Nazwę gałęzi śledzonej odpowiadającej gałęzi `X` poznamy także, wydając komendy:

```
git config --get branch.X.remote  
git config --get branch.X.merge
```

Do ręcznego ustalenia gałęzi śledzonej możemy użyć polecenia:

```
git branch --set-upstream gałaz-lokalna repozytorium-zdalne/gałaz-zdalna
```

Polecenie:

```
git branch --set-upstream master origin/master
```

ustala, że gałęzią śledzoną dla gałęzi `master` będzie gałąź `master` w repozytorium `origin`.

Podobnie polecenie:

```
git branch --set-upstream lorem ipsum/dolor
```

ustala, że gałęzią śledzoną dla gałęzi `lorem` będzie gałąź `dolor` w repozytorium o nazwie `ipsum`.

Wydruk generowany poleceniem:

```
git config --list
```

przyjmie postać:

```
branch.lorem.remote=ipsum  
branch.lorem.merge=refs/heads/dolor
```



Wskazówka

Polecenie:

```
git branch --set-upstream master origin/master
```

jest równoważne dwóm poleceniom:

```
git config branch.master.remote origin  
git config branch.master.merge refs/heads/master
```

Listowanie gałęzi

Do sprawdzania listy gałęzi lokalnych służy poznane w części drugiej polecenie:

```
git branch
```

Gałęzie zdalne poznamy, wydając polecenie:

```
git branch -r
```

Komenda:

```
git branch -a
```

wyświetla listę wszystkich gałęzi.

Skorowidz

B

Bazaar, 270
baza danych
 repozytorium, *Patrz:* repozytorium
 rewizji, *Patrz:* repozytorium
Bitbucket, 20, 269, 270, 280, 283, 287, 292, 293, 313
 interfejs, 275
branch, *Patrz:* gałąź

C

commit, *Patrz:* rewizja, operacja zatwierdzenia
Cygwin, 278
cytowanie, 102

D

diagram stanów, 78
DVCS, *Patrz:* system kontroli wersji
dziennik reflog, 100, 137, 154, 161, 162, 178, 237

E

edytor
 tekstowy, 28, 58
 vi, 28, 59

F

fast forward, *Patrz:* przewijanie do przodu
file
 added, *Patrz:* plik dodany
 deleted, *Patrz:* plik usunięty

modified, *Patrz:* plik zmodyfikowany
renamed, *Patrz:* plik o zmienionej nazwie
staged, *Patrz:* plik indeksowany
unmodified, *Patrz:* plik aktualny
unstaged, *Patrz:* plik niezaindeksowany
untracked, *Patrz:* plik nieśledzony
file modified, *Patrz:* plik zmodyfikowany
folder
 domowy użytkownika, 107
 roboczy, 43
fork, *Patrz:* rozgałęzianie

G

gałąź, 141, 156, 161, 181, 303
 bieżąca, 142, 148, 153, 193
 lista, 194
 lokalna, 194, 217, 285
 łączenie, 167, 170, 171, 175, 176, 178, 181, 195, 233
 master, 141, 153, 211, 215, 285
 przełączanie, 145, 147, 208
 przesyłanie ze zmianą nazwy, 209
 przesyłanie, 210
 rozłączność, 150
 śledzona, 193, 194, 199, 209
 tworzenie, 141, 143, 144, 147
 tymczasowa, 221
 usuwanie, 141, 153, 154, 157, 209
 wysyłanie, 206
 zawieranie, 150
 zdalna, 194, 209
 zmiana nazwy, 155
GFM, 313
Github, 19, 20, 269, 279, 283, 286, 292, 293, 313
 interfejs, 271

Github Flavored Markdown, *Patrz*: GFM
 GitPad, 28
 Google Code, 19
 graf niespójny, 223

H

HEAD, 93, 96
 hosting, 19

I

importowanie kodu, 286
 indeks, 56, 72, 127, 128, 198, 245
 indeksowanie, 52

J

język
 bash, 109
 Markdown, 313
 jQuery, 39

K

klient
 Git, 278
 github.com, 277
 klucz SSH, 278, 279, 280
 komenda, 130
 echo, 135
 find, 24, 131, 134
 git, 24, 130
 git add, 43, 51, 57, 75, 128, 130, 133, 134
 git archive, 89, 291
 git backup, 210
 git branch, 46, 130, 133, 143, 144, 153, 155,
 182, 189, 194, 226, 227
 git checkout, 45, 86, 125, 133, 145, 147, 148,
 164, 182, 208, 217, 229, 241, 242, 243, 260,
 264
 git checkout master, 46
 git clone, 30, 129, 131, 187, 191
 git commit, 43, 58, 75, 116, 117, 128, 133,
 134, 162, 199, 260
 git config, 130, 183, 193, 227, 259, 265
 git diff, 157, 183, 245, 248, 263, 265
 git fetch, 188, 193, 195, 226, 227
 git gc, 183
 git gui, 132
 git help add, 130
 git help branch, 130
 git help config, 130

git help init, 130
 git init, 29, 130, 131, 141, 188
 git log, 33, 91, 96, 104, 131, 132, 173, 255,
 263, 265
 git merge, 168, 170, 171, 173, 176, 178, 181,
 183, 195, 217, 228, 233, 238, 239
 git mv, 60
 git prune, 164, 183
 git pull, 72, 129, 193, 196, 198, 199, 228, 301
 git push, 72, 129, 193, 199, 210, 217, 228, 301
 git rebase, 116, 117, 119, 175, 176, 178, 181,
 184, 217, 236, 238, 240, 264
 git reflog, 100, 137, 184
 git remote, 227
 git remote add, 192, 222, 225, 227, 301
 git remote add origin, 188, 191, 219, 220, 226
 git remote rm, 227
 git reset, 39, 86, 116, 133, 157, 173, 178, 184,
 189, 226
 git revert, 116, 120, 127, 181
 git rev-list, 137
 git rev-parse, 101, 137
 git rm, 51, 59, 134
 git shortlog, 131
 git show, 85, 242, 243, 264
 git simple commits, 121
 git simple-commit, 110, 112
 git simple-loop, 111
 git status, 68, 75, 128, 133, 182
 git symbolic, 182
 git tag, 84
 gitk, 132
 skrót, 107
 ssh, 24
 wc, 24
 konflikt, 233, 236, 263
 binarny, 238, 242, 263
 dublowanie, 238
 tekstowy, 233, 236, 242, 263
 konsola, 25, 26
 bash, 26
 kontekst, 248
 kontrola
 akceptowanych rewizji, 269
 spójności danych, 129
 uprawnień użytkowników, 269
 kopia bezpieczeństwa, 30, 291

L

Linux, 278
 lokalność, 129

M

menu kontekstowe, 26
 Mercurial, 270
 migawka, *Patrz: snapshot*

N

nazwa symboliczna HEAD, 93, 96

O

obszar roboczy, 39, 56, 72, 127, 128, 132, 148,
 198, 211, 245
 openssh, 278
 operacja zatwierdzania, 15, 16

P

pakiet openssh, 278
 parent, *Patrz: rodzic*
 plik

- aktualny, 51, 52, 54, 128
- binarny, 257
- dodany, 69
- git/HEAD, 93
- git/info/exclude, 75, 76, 135
- gitattributes, 258, 261
- gitignore, 75, 76, 135
- ignorowany, 53, 75, 78, 128
- konfiguracyjny, 76, 131
- konfiguracyjny gitconfig, 107
- nieignorowany, 53, 78, 128
- nieśledzony, 51, 52, 53, 57
- niezaindeksowany, 53, 54, 56, 59, 128
- o zmienionej nazwie, 69
- odpowiadające rewizji, 89
- przywracanie, 40
- stan, 68, 127
- stan dwuliterowy, 69
- tekstowy, 238, 257
- usunięty, *Patrz: plik usunięty*
- zaindeksowany, 53, 56, 128
- zmiana nazwy, 60
- zmieniony, 255
- zmodyfikowany, 51, 52, 69

 polecenie, *Patrz: komenda*
 praca grupowa, 18, 30, 127, 233, 291, 293
 program GitPad, 28
 projekt

- historia, 18, 33, 36, 115, 116, 125, 127
- hosting, 19
- stan, 17, 108, 125

protokół

- file, 277
- Git, 277
- HTTPS, 277
- SSH, 277, 280

 przestrzeń robocza, 51, 52
 przewijanie do przodu, 168, 169, 199, 223
 przodek, 96
 pull request, *Patrz: żądanie aktualizacji*

R

repo, *Patrz: repozytorium*
 repository, *Patrz: repozytorium*
 bare, *Patrz: repozytorium surowe*
 repozytorium, 14, 18, 29, 30, 39, 48, 52, 56, 127,
 128, 141

- główne, 216
- inicjalizowanie, 29
- klonowanie, 30, 187, 188, 191, 293, 301, 302
- lokalne, 187, 188, 195, 198, 199, 225, 302
- łączenie, 219, 223
- o historii nieliniowej, 94
- prywatne, 271, 291, 292
- publiczne, 271
- stan, 61
- surowe, 72, 198, 216, 226
- synchronizacja, 195, 225
- śledzone, 272
- tworzenie, 283, 287
- uaktualnianie, 197, 199
- współdzielenie, 269, 293
- zdalne, 187, 188, 192, 193, 195, 199, 209,
 211, 219, 225, 280
- zwykłe, 72, 210

 revision, *Patrz: rewizja*
 revision control system, *Patrz: system kontroli
 wersji*
 rewizja, 16, 18, 34, 45, 52, 127, 156, 245

- akceptowana, 269
- domyślna, 93
- graf niespójny, 223
- identyfikacja, 91, 92, 93, 100
- identyfikator, 34
- łączenie, 115, 117, 119
- tworzenie, 58, 142, 143
- usuwanie, 115, 116, 127
- zabezpieczanie przed utratą, 209
- zgubiona, 149, 163

 rodzic, 95, 96, 97
 rozgałęzianie, 293

S

serwer
 bitbucket.org, 25
 SSH, 25
 skrót SHA-1, 91, 92, 96, 129, 148, 154, 164, 195
 snapshot, 44
 Source Forge, 19
 stan detached HEAD, 148, 163, 164, 181, 236, 237
 stan projektu, *Patrz:* projekt stan
 strumień przekierowanie, 102
 system
 jądro, 13
 kontroli rewizji, *Patrz:* system kontroli wersji
 kontroli wersji, 13, 270
 śledzenia błędów, 30, *Patrz:* śledzenie błędów

Ś

ścieżka dostępu, 24
 śledzenie błędów, 269, 291, 315, 319

T

tag, *Patrz:* znacznik
 annotated, *Patrz:* znacznik opisany
 lightweight, *Patrz:* znacznik lekki

U

ujednolicony format opisu, 246

V

vi, 28, 59

W

wiersz poleceń, 26, 102
 working area, *Patrz:* obszar roboczy
 working directory, *Patrz:* obszar roboczy

Z

zmienna
 środowiskowa
 PATH, 23
 znacznik, 83, 92, 96, 156, 291
 dane, 85
 dostępność, 85, 136
 konfliktu, 234
 lekki, 83, 84
 opisany, 83, 84
 tworzenie, 136
 usuwanie, 85, 136
 znak
 !, 108
 ", 102, 103
 ^, 97, 98, 99, 102
 |, 102
 <>, 102
 >, 102
 końca wiersza, 259, 260, 261
 tylda, 96, 98, 99
 złamania wiersza, 24

Ż

żądanie aktualizacji, 269, 293, 294, 301, 303, 319

PROGRAM PARTNERSKI

— GRUPY HELION —

- 
1. ZAREJESTRUJ SIĘ
 2. PREZENTUJ KSIĄŻKI
 3. ZBIERAJ PROWIZJĘ

Zmień swoją stronę WWW w działający bankomat!

Dowiedz się więcej i dołącz już dzisiaj!

<http://program-partnerski.helion.pl>

GRUPA
Helion

Git – oszczędź sobie kłopotów z synchronizacją projektu!

Praca nad niemal każdym projektem informatycznym wymaga współdziałania wielu osób, często pracujących z dala od siebie. W takich warunkach bardzo łatwo popełnić błąd, napisać jakiś ważny plik albo przypadkowo zdublować dane. Mały projekt po takiej wpadce da się jeszcze uratować, ale większy można wyrzucić do kosza. Chyba że od momentu jego inicjalizacji używamy narzędzia odpowiedzialnego za właściwą synchronizację danych, czyli systemu kontroli wersji, co jest standardem we współczesnej informatyce. Jednym z takich programów jest git, napisany na potrzeby zarządzania kodem źródłowym jądra systemu Linux – taka rekomendacja mówi sama za siebie.

Możliwości programu git i sposoby jego praktycznego zastosowania w różnych projektach przedstawione zostały w tej książce. Znajdziesz tu podstawowe informacje o instalacji środowiska i tworzeniu repozytoriów, pracy z plikami, identyfikowaniu rewizji i zmienianiu historii projektu. Dowiesz się, kiedy i jak tworzyć czy łączyć gałęzie oraz całe repozytoria, korzystać z repozytorium lokalnego i zdalnego, a także synchronizować je w odpowiedni sposób. Poznasz możliwe konflikty między wersjami pliku i nauczysz się radzić sobie z nimi. Zrozumiesz, jak wykorzystywać najbardziej znane serwery hostingowe dla projektów git oraz dostępne w nich wbudowane systemy śledzenia błędów. I wreszcie przestaniesz mieć koszmary, w których tracisz tygodnie na odszukanie zagubionego pliku. Git zrobi to za Ciebie!

- Instalacja programu git, tworzenie repozytoriów i obszar roboczy
- Tworzenie rewizji i przywracanie stanu plików
- Stany plików, ignorowanie plików i znaczniki
- Identyfikowanie rewizji, skróty komend
- Modyfikowanie historii projektu oraz tworzenie i usuwanie gałęzi
- Łączenie gałęzi: operacja merge i operacja rebase
- Powiązanie repozytorium lokalnego i zdalnego oraz podstawy synchronizacji repozytoriów
- Praktyczne wykorzystanie git i łączenie oddzielnych repozytoriów
- Treść pliku: konflikty, badanie różnic, pliki tekstowe i binarne
- Serwisy github.com i bitbucket.org
- Praca grupowa w serwisach github.com oraz bitbucket.org i zintegrowany system śledzenia błędów

Wypróbuj git – wystarczająco dobry nawet dla jądra Linuksa!

	<i>Sprawdź nasze szkolenia!</i>	KOD KORZYŚCI <i>Sięgnij po więcej!</i>	
 helion.pl	 AKADEMIA IT & BUSINESS	ISBN 978-83-283-9520-6	
 HELION SA ul. Kościuszki 1c 44-100 Gliwice tel.: 32 230 99 63 helion@helion.pl	HELIONSZKOLENIA.PL	9 788328 395206	
INFORMATYKA W NAJLEPSZYM WYDANIU		Cena: 69,00 zł	