Helion

Git

Włodzimierz Gajda

Rozproszony system kontroli wersji

Redaktor prowadzący: Ewelina Burska
Projekt okładki: Jan Paluch
Materiały graficzne na okładce zostały wykorzystane za zgodą Shutterstock.

Wydawnictwo HELION
ul. Kościuszki 1c, 44-100 GLIWICE
tel. 32 231 22 19, 32 230 98 63
e-mail: helion@helion.pl
WWW: http://helion.pl (księgarnia internetowa, katalog książek)

Drogi Czytelniku!
Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres
http://helion.pl/user/opinie?gitroz
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

ISBN: 978-83-246-5564-9

Printed in Poland.

Spis treści

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**.

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.

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.:

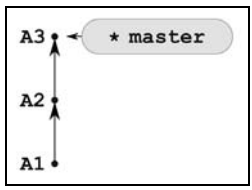
```
git remote add origin C:\my\repos\example
```

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=...
```

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 galaz-lokalna repozytorium-zdalne/galaz-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
```

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

M

N

O

P

R

S

Ś

T

U

V

W

Z

Ż

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, nadpisać 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 miewać 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!

Nr katalogowy: 11471

Helion SA
ul. Kościuszki 1c, 44-100 Gliwice
tel.: 32 230 98 63
e-mail: helion@helion.pl
http://helion.pl

Cena 54,90 zł
ISBN 978-83-246-5564-9
9 788324 655649

Informatyka w najlepszym wydaniu