Git

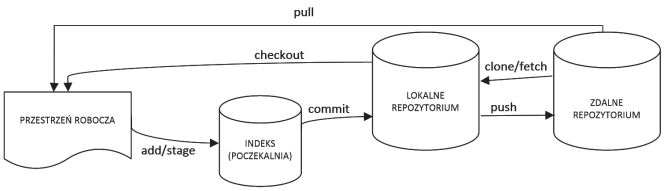
TABLICE INFORMATYCZNE • Daniel Krasnokucki

Helion

WPROWADZENIE

Przepływ

Pracując w zespole lub nad dużym projektem, chcemy mieć porządek w repozytorium kodu. Git bardzo ułatwia zapanowanie nad poszczególnymi wersjami, historią i współpracą przy tworzeniu oprogramowania. Każdy z użytkowników będzie pracował w swojej przestrzeni roboczej (pliki będzie zmieniał lokalnie), a dopiero po dodaniu zmian do indeksu i zatwierdzeniu ich do repozytorium lokalnego można wrzucić pliki na serwer. Podstawowy przepływ plików w Gicie pokazany został poniżej na rysunku.



Słownik

branch — odgałęzienie.
commit — stan projektu (migawka) w danej chwili.
indeks (ang. *staging area*) — poczekalnia. Trzyma zmiany, które zostaną zacommitowane do nowej wersji.
HEAD — wskaźnik do gałęzi, w której się obecnie znajdujesz.
katalog roboczy (ang. *working directory*) — katalog, w którym dokonujemy zmian.
master — zwyczajowa nazwa głównego odgałęzienia.
merge — scalanie/łączenie plików lub gałęzi w jedną.
obiekt — coś, co jest w repozytorium (obiektami są: blob, drzewo, commit, tag).
origin — zwyczajowa nazwa głównego repozytorium.
nazwa — nazwa obiektu, dwudziestocyfrowy skrót SHA-1 dla danego obiektu.
ORIG_HEAD — wskaźnik do obecnej gałęzi repozytorium zdalnego.
ref — referencja, wskaźnik na commit (może to być gałąź, tag itd.).
repozytorium — struktura danych zawierająca historię projektu.
repozytorium lokalne — miejsce przechowywania zacommitowanych plików.
repozytorium zdalne — wersja projektu utrzymywana na serwerze.
revision — rewizja, kolejna wersja plików w repozytorium.
tag — zrozumiała dla człowieka nazwa obiektu wraz z informacją o osobie tagującej.

PODSTAWOWE POLECENIA

Pomoc

`--help` — wyświetlenie pomocy dla dowolnego polecenia.

Przykład: `git config --help`

Konfiguracja repozytorium

`git init` — inicjalizacja repozytorium, czyli stworzenie w miejscu wywołania podkatalogu *.git* zawierającego szkielet repozytorium.
`git init --bare` — inicjalizacja repozytorium zdalnego w obecnym katalogu.
`git init <TwojaLokalizacja>` — inicjalizacja repozytorium w podanej lokalizacji.

Przykład: `git init C:\Git\TwojeRepozytorium\`

`git config` — zmiana ustawień konfiguracyjnych.
`--global` — zmiana ustawień globalnych.
`--local` — zmiana ustawień lokalnych.
`--system` — zmiana ustawień systemowych (dla wszystkich użytkowników).
`--list` — wyświetlenie obecnych ustawień.

Więcej opcji ustawień w części „Konfiguracja".

`git remote add origin https://TwojAdres/twoje_repozytorium.git` — ustawienie zdalnego repozytorium.
`git clone <zdalne_repozytorium> <lokalny_katalog>` — klon repozytorium zdalnego do katalogu lokalnego. Automatycznie ustawia *origin* na klonowane repozytorium zdalne.

Operacje w repozytorium

`git add` — dodanie plików do indeksu (pliki będą gotowe do commita).
`-a` — dodaje wszystkie pliki (nowe, zmodyfikowane, usunięte).
`.` — dodaje nowe i zmodyfikowane pliki (bez usuniętych).
`-u` — dodaje pliki zmodyfikowane i usunięte (bez nowych).
`git commit` — zatwierdzenie zmian w indeksie, dodanie do lokalnego repozytorium.

Przykład: `git commit -m "Opis zmian"`

`git mv` — przeniesienie lub zmiana nazwy pliku i zapis w indeksie.
`git rm` — usuwa pliki z katalogu roboczego.
`git status` — wyświetla informację o tym, które zmiany w katalogu roboczym zostały zapisane w indeksie, a które nie.
`-s` — wyświetla krótszy opis.

Niektóre statusy:

STATUS	SKRÓT	OPIS
untracked	??	plik nieśledzony przez Gita/niewersjonowany
modified	M	zmieniony plik
added	A	plik dodany
deleted	D	plik usunięty
copied	C	plik skopiowany
renamed	R	nazwa pliku została zmieniona
updated but unmerged	U	pliki zmienione, ale niescalone

Poprawki

`git checkout -- <nazwa_pliku>` — przywrócenie pliku do wersji z dowolnego commita.
`git reset` — usunięcie plików z indeksu, ustawienie bieżącej wersji (`HEAD`). Pliki w katalogu roboczym nie zmieniają się.
`git reset --hard` — skasowanie wszystkich zmian, przywrócenie stanu z ostatniej rewizji.
`git reset --soft` — skasowanie commitów, bez zmian w indeksie i katalogu roboczym.
`git commit --amend` — wprowadzenie zmian do ostatniej rewizji (poprawienie ostatniego commita).

Praca z gałęziami

`git branch` — wyświetla gałęzie w lokalnym repozytorium.
`git branch <nazwa>` — tworzy nowe odgałęzienie.
`git branch -d <nazwa>` — usuwa gałąź o podanej nazwie. Wersja z tej gałęzi będzie jednak nadal dostępna z innych gałęzi.
`git checkout` — przywraca pliki do stanu z ostatniego commita.
`git checkout <nazwa_gałęzi>` — zmienia aktywną gałąź na wybraną przez użytkownika.
`git checkout -b <nazwa_gałęzi>` — tworzy nową gałąź na podstawie aktywnej i przełącza się na nią.
`git merge` — scalanie dwóch lub więcej wersji z projektem.
`git merge --no-ff` — scalenie zmian ze szczegółowym opisem w specjalnie utworzonym commicie.
`git mergetool` — uruchamia narzędzie do scalania plików.
`git stash` — zapamięta wszelkie niezatwierdzone zmiany i przywróci stan kopii roboczej do ostatniego commita (ukryje nasze zmiany w skrytce).
`git stash apply` — polecenie odwrotne do poprzedniego. Odkrywa schowane pliki.
`git stash pop` — przywrócenie ze skrytki *stash* poprzedniego stanu katalogu roboczego i indeksu.
`git stash drop` — usunięcie zapamiętanego stanu ze stosu skrytki.
`git stash show` — wyświetlenie zmian w indeksie oraz w plikach w najnowszym wpisie w skrytce.
`git stash list` — wyświetlenie stosu przechowywanych w skrytce kontekstów, zaczynając od najnowszego.
`git tag` — wyświetla wszystkie dostępne etykiety.

Przykład: `git tag -l 'v1.1.2.*'` — wyszukiwanie etykiet wersji 1.1.2.

Aktualizacja projektu

`git fetch` — pobieranie zmian z odpowiednich gałęzi zdalnych do gałęzi lokalnych je śledzących bez scalania zmian.
`git pull` — pobranie zmian ze zdalnego repozytorium do siebie. Próbuje automatycznie scalić (wmergować) zmiany.
`git push` — zapis śledzonych gałęzi do zdalnego repozytorium.
`git push -f` — wysyła zmiany do zdalnego repozytorium, ignorując konflikty; oznacza to, że jeśli wystąpią konflikty, pliki zostaną nadpisane obecną wersją.
`git push [remote-name] [branch-name]` — zapisuje konkretne odgałęzienie w repozytorium zdalnym.

Przykład: `git push origin master`

`git remote` — wyświetlanie listy repozytoriów zdalnych.
`git submodule add` — dodanie zewnętrznego projektu jako modułu zależnego.

Sprawdzanie i porównywanie zmian

`git describe` — znajduje ostatnią etykietę dostępną dla commita.
`git diff` — wyświetlenie różnic pomiędzy indeksem a katalogiem roboczym.
`git diff <nazwa pliku>` — szczegółowe wyświetlenie zmian w wybranym pliku.
`git log` — wyświetlenie historii commitów do bieżącej wersji.
`git show` — wypisanie zawartości obiektów dowolnego typu.
`git shortlog` — lista osób wrzucających paczki, wraz z wylistowanymi tytułami ich commitów.

Patche, łaty, wprowadzanie zmian

`git apply` — zastosowanie patcha, który otrzymaliśmy, lub różnic, które znaleźliśmy.

Przykład: `git apply naprawiony_blad_w_aplikacji.patch`

`git apply --check` — sprawdzenie, czy patch można zastosować bez konfliktów.

`git am` — zastosowanie do bieżącego repozytorium podanej łaty lub wielu łat.

Przykład: `git am *.patch`

`git cherry-pick` — wprowadzenie pojedynczej zmiany z dowolnej gałęzi do tej, na której obecnie pracujesz.
`git rebase` — przeniesienie commita z jednej gałęzi na drugą.
`git revert` — wymazywanie zmian z podanego commita.

POLECENIA DLA ZAAWANSOWANYCH

Debugowanie

`git bisect` — narzędzie, które pozwala na przeszukiwanie repozytorium metodą binarną.
`.git bisect reset` — powrót do miejsca, gdzie byliśmy przed przeszukiwaniem.
`git blame <nazwa_pliku>` — komentuje każdą linię podanego pliku, by pokazać, kto i kiedy ją ostatnio zmieniał.
`git grep wyrażenie_regularne <nazwa_pliku>` — wyszukiwanie w pliku napisów spełniających wyrażenie regularne i wyświetlenie ich.

Administracja

`git archive` — eksportowanie danych (plików, gałęzi, commita).
`git gc` — czyszczenie niepotrzebnych plików i optymalizacja repozytorium lokalnego.
`git fsck` — narzędzie do sprawdzania zawartości bazy pod względem integralności danych.
`git reflog` — pozwala zobaczyć, na jakim etapie był projekt w każdym momencie (pokazuje także SHA).
`git filter-branch` — pozwala na nadpisanie wielu commitów historycznych naraz.
`git bundle` — pozwala przenosić obiekty i referencje jako archiwum.
`git instaweb` — narzędzie pozwalające przeglądać repozytorium w przeglądarce internetowej.
`git clean` — usuwanie niewersjonowanych plików z przestrzeni roboczej.
`git prune` — usuwanie rewizji, które zostały zagubione.
`gitk` — narzędzie do przeglądania repozytorium.

Polecenia instalatorskie (plumbing commands)

`git ls-tree` — wylistowanie zawartości obiektu (np. rewizji, folderu).
`git cat-file` — pobranie danych z Gita.
`git commit-tree` — stworzenie commita na podstawie dostępnego elementu (np. katalogu).
`git count-objects` — zliczenie niespakowanych obiektów i zajmowanego przez nie miejsca.
`git diff-index` — porównanie podanego obiektu z aktualnym lub dwóch obiektów, do których ścieżkę podano.
`git for-each-ref` — wyświetlenie wszystkich dostępnych referencji (*refs*).
`git for-each-ref <pattern>` — wyświetlenie wszystkich referencji (*refs*) spełniających podany wzorzec.
`git hash-object` — zapisanie nowego obiektu binarnego i zwrócenie jego sumy SHA-1.
`git ls-files` — wylistowanie plików.

Przykład: `git ls-files -m` — wyświetlenie wszystkich zmodyfikowanych plików.

`git merge-base` — znalezienie najlepszego dopasowania (bazy łączenia/mergowania) pomiędzy dwoma odgałęzieniami.
`git read-tree` — wczytywanie obiektu drzewa do indeksu.
`git rev-list` — wyświetlenie listy commitów w kolejności chronologicznej od najmłodszych.
`git rev-parse <przedrostek>` — wyszukuje hash obiektu o podanym przedrostku.

Przykład: `git rev-parse 812922c412`

`git show-ref` — wylistowanie wszystkich referencji w lokalnym repozytorium.
`git symbolic-ref <nazwa>` — wyświetlenie lub zmiana symbolicznej nazwy gałęzi.

Przykład: `git symbolic-ref HEAD` może wyświetlić nam: *refs/heads/master*.

`git symbolic-ref <nazwa> --delete` — usunięcie wybranej nazwy symbolicznej gałęzi.
`git update-index` — modyfikuje indeks, aktualizując wersje wskazanych plików z drzewa roboczego.
`--add` — jeśli pliku nie ma w indeksie, jest dodawany (domyślnie jest pomijany).
`--remove` — jeśli plik jest w indeksie, ale nie ma go w katalogu roboczym, jest usuwany z indeksu (domyślnie jest pomijany).
`--assume-unchanged <ścieżka>` — pozwala zaprzestać wyświetlania i śledzenia zmian w podanym katalogu lub pliku wcześniej stworzonym (np. jakbyśmy chcieli lokalnie trzymać użytkownika i hasło do profilu, ale nie wrzucać go do zdalnego repozytorium).
`git update-ref` — aktualizacja nazwy obiektu przechowywanego w referencji.

Przykład: `git update-ref HEAD <nowa_nazwa>`

`git verify-pack` — walidacja spakowanych archiwów gitowych.
`git write-tree` — stworzenie obiektu drzewa na podstawie obecnego indeksu (indeks musi być w pełni scalony).

Konfiguracja

`git --version` — sprawdzenie wersji gita, która jest używana.

Przykłady konfiguracji z użyciem `git config`:

`git config --global user.email "twoj@email.com"` — ustawienie e-maila użytkownika.
`git config --global user.name "Twoje Imie"` — ustawienie nazwy użytkownika.
`git config --global sendemail.smtpserverport 465` — ustawienie portu dla SMTP.
`git config --global core.editor <ścieżka_do_edytora>` — ustawienie domyślnego edytora plików.
`git config --global http.sslVerify false` — ustawienie braku weryfikacji certyfikatu SSL podczas połączenia po HTTP.

Inne przydatne polecenia i informacje

`git rev-parse <nazwa_rewizji>` — znajduje klucz SHA-1 podanej rewizji.

SSH

Pracując w zespole, najprawdopodobniej będziesz potrzebował publicznego klucza ssh. Aby go wygenerować, użyj polecenia:

```
ssh-keygen -C "twój@email.com" -t rsa
```

Klonowanie zdalnego repozytorium z użyciem protokołu SSH:

```
git clone ssh://user@server/project.git
```

.GITIGNORE

W pliku *.gitignore* podajemy pliki lub wzorzec dla plików, które chcemy pomijać przy dodawaniu do repozytorium. Można go utworzyć w dowolnym katalogu i będzie miał on zastosowanie do katalogu, w którym jest umieszczony, oraz wszystkich jego podkatalogów.
Każdy wzorzec lub plik powinien być umieszczony w nowej linii.
Komentarze zaczynamy znakiem `#`.
Wykrzyknik na początku wiersza oznacza zanegowanie reguły po nim następującej. Ma on priorytet przed innymi regułami.

ALIAS

Aby ułatwić sobie pracę, można skorzystać z opcji aliasowania w Gicie. Można to zrobić na 2 sposoby:

1. Dodać wpis z aliasami do pliku *.gitconfig* katalogu użytkownika:

```
[alias]
 co = checkout
 ci = commit
 st = status
 br = branch
```

2. Przypisać aliasy dla najczęściej używanych poleceń w ustawieniach, np.:

```
git config --global alias.co checkout
git config --global alias.br branch
git config --global alias.ci commit
git config --global alias.st status
```

`git config --global alias.co checkout` — ustawia alias `co` dla polecenia `checkout`

ISBN: 978-83-283-4573-7

Wydawnictwo Helion
ul. Kościuszki 1c, 44-100 Gliwice
tel. +48 32 230 98 63
e-mail: helion@helion.pl
www.helion.pl