

Praca na repozytoriach i gałęzie

AM

29 września 2024



1 Repozytoria z zewnętrznymi źródłami

Praca z systemem Git odbywa się z reguły z użyciem zdalnych repozytoriów znajdujących się na zewnętrznych serwerach. Aby wyświetlić listę wszystkich zdalnych repozytoriów powiązanych z lokalnym repozytorium w aktywnym katalogu używamy komendy:

```
git remote -v
```

Pozwoli to zobaczyć listę wszystkich repozytoriów wraz z ich źródłami dla pobierania (fetch) i wysyłania (push) migawek.

Wspomniana wcześniej komenda "git clone" powoduje utworzenie wpisu na liście oznaczonego jako "origin". Chcąc zmienić etykietę repozytorium używamy komendy:

```
git remote rename [stara etykieta] [nowa etykieta]
```

Przykładowo, chcąc zmienić etykietę "origin" na "repo" polecenie będzie wyglądać następująco:

```
git remote rename origin repo
```

Jeżeli chcemy dodać inne zdalne repozytorium używamy polecenia:

```
git remote add [etykieta] [URL]
```

Jeżeli chcemy uzyskać informacje dotyczące zdalnego repozytorium przed jego dodaniem używamy komendy:

```
git remote show [URL]
```

Możemy również uzyskać informacje o repozytorium poprzez podanie jego etykiety, jeżeli takowe już zostało dodane do naszej listy. Wtedy polecenie będzie różnić się jedynie argumentem:

```
git remote show [etykieta]
```

Jeżeli zaistnieje potrzeba usunięcia zdalnego repozytorium dokonać tego możemy przy pomocy polecenia:

```
git remote rm [etykieta]
```

Po dodaniu nowego zdalnego repozytorium możemy pobrać jego aktualny stan za pomocą komendy:

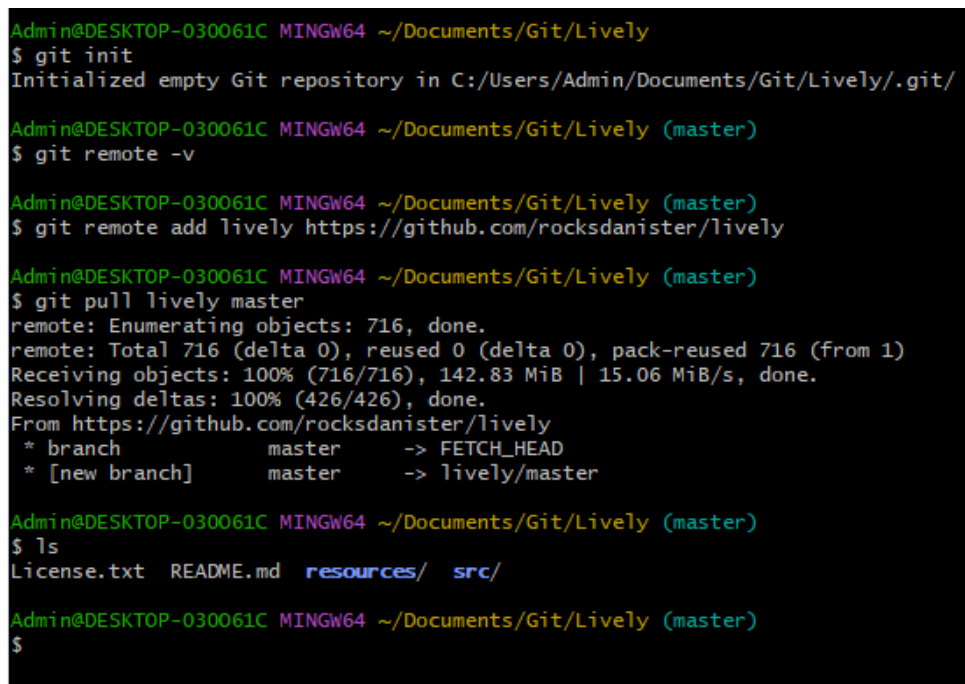
```
git pull [etykieta] [gałąź]
```

Przykładowo, aby pobrać stan zdalnego repozytorium z etykietą "repo" i gałęzią główną "main" do repozytorium lokalnego użyjemy komendy:

```
git pull repo main
```

Jeżeli chcemy wysłać nową wersję repozytorium na serwer zewnętrzny musimy użyć komendy:

```
git push [etykieta] [gałąź]
```



```
Admin@DESKTOP-030061C MINGW64 ~/Documents/Git/Lively
$ git init
Initialized empty Git repository in C:/Users/Admin/Documents/Git/Lively/.git/

Admin@DESKTOP-030061C MINGW64 ~/Documents/Git/Lively (master)
$ git remote -v

Admin@DESKTOP-030061C MINGW64 ~/Documents/Git/Lively (master)
$ git remote add lively https://github.com/rocksdanister/lively

Admin@DESKTOP-030061C MINGW64 ~/Documents/Git/Lively (master)
$ git pull lively master
remote: Enumerating objects: 716, done.
remote: Total 716 (delta 0), reused 0 (delta 0), pack-reused 716 (from 1)
Receiving objects: 100% (716/716), 142.83 MiB | 15.06 MiB/s, done.
Resolving deltas: 100% (426/426), done.
From https://github.com/rocksdanister/lively
 * branch          master       -> FETCH_HEAD
 * [new branch]    master       -> lively/master

Admin@DESKTOP-030061C MINGW64 ~/Documents/Git/Lively (master)
$ ls
License.txt  README.md  resources/  src/

Admin@DESKTOP-030061C MINGW64 ~/Documents/Git/Lively (master)
$
```

Rysunek 1: Przykład zainicjowania nowego repozytorium lokalnego, dodania zdalnego repozytorium i pobraniu plików z aktualnej wersji repozytorium zdalnego do katalogu roboczego z podaną gałęzią główną "master".

2 Gałęzie

Git umożliwia tworzenie branch'y, będących gałęziami głównego "pnia" repozytorium. Oznacza to, że kiedy na przykład wprowadzić chcemy nową funkcjonalność do naszego projektu, lub zwyczajnie zacząć eksperymentować, możemy utworzyć nową gałąź i pracować nad projektem, bez ryzyka uszkodzenia już działającego kodu. Jeżeli wszystko zostanie zrealizowane pomyślnie można scalić gałąź eksperymentalną z gałęzią główną implementując zmiany w głównym pniu repozytorium. Aby utworzyć nową gałąź użyjemy komendy:

```
git branch [gałąź]
```

...gdzie "gałąź" to nazwa gałęzi.

Aby przejść do pracy nad konkretną gałęzią używamy polecenia:

```
git checkout [gałąź]
```

Komenda `git checkout` może również zostać użyta do przejścia do jakiegokolwiek migawki repozytorium, zamieniając zawartość katalogu roboczego na zawartość z migawki. Po przejściu do migawki nie będziemy pracowali na jakiegokolwiek gałęzi. Oznacza to, że zatwierdzone zmiany w plikach będą w swego rodzaju "limbo", o ile nie utworzymy nowej gałęzi aby je zapisać.

Jeżeli scalić chcemy jedną z naszych gałęzi z gałęzią główną musimy najpierw przejść do pracy nad gałęzią główną:

```
git checkout master
```

...a następnie wprowadzić polecenie:

```
git merge [gałąź]
```

Podczas scalania gałęzi może jednak dojść do konfliktu jeżeli w dwóch gałęziach istnieją dwie zupełnie różne wersje pliku. Przykładowo: jeżeli rozpoczęte zostały prace nad gałęzią "A", w celu zaimplementowania nowej funkcji poprzez edycję pliku "index.html", a w międzyczasie w gałęzi głównej naprawiony został krytyczny błąd w pliku "index.html", to przy scalaniu gałęzi "A" z gałęzią główną może dojść do konfliktu, ponieważ istnieją dwie zupełnie różne wersje tego samego pliku. W takim przypadku musimy otworzyć plik i manualnie rozstrzygnąć, co chcemy zachować, a co nie. Po rozstrzygnięciu konfliktu wpisujemy polecenie `git add index.html` i zatwierdzamy zmiany.

3 Tagi

Tagi umożliwiają zaznaczenie ważnych wersji projektu poprzez dodanie etykiety do konkretnej migawki (np. v1.0). W Git istnieją dwa rodzaje tagów - tagi lekkie i tagi opisane.

Tagi lekkie to tagi, które są jedynie wskaźnikami do konkretnych migawek z dopisaną etykietą. Po dodaniu tagu lekkiego do migawki będzie ona oznaczona podaną przez nas etykietą, ale po zatwierdzeniu nowych zmian etykieta nie przeniesie się do następnej migawki. Aby dodać tag lekki do aktualnej migawki używamy komendy:

```
git tag [etykieta]
```

Tagi opisane są przechowywane jako pełne obiekty w bazie danych Git'a i zachowują dodatkowe metadane o autorze migawki, dacie jej utworzenia oraz zawierają dołączoną wiadomość. Ze względu na ilość informacji dołączonych do tagów opisowych są one zalecane w przypadku głównych wersji projektu. Aby dodać tag opisowy do aktualnej migawki używamy komendy "git tag" z dołączonymi parametrami "a" oraz "m" w następujący sposób:

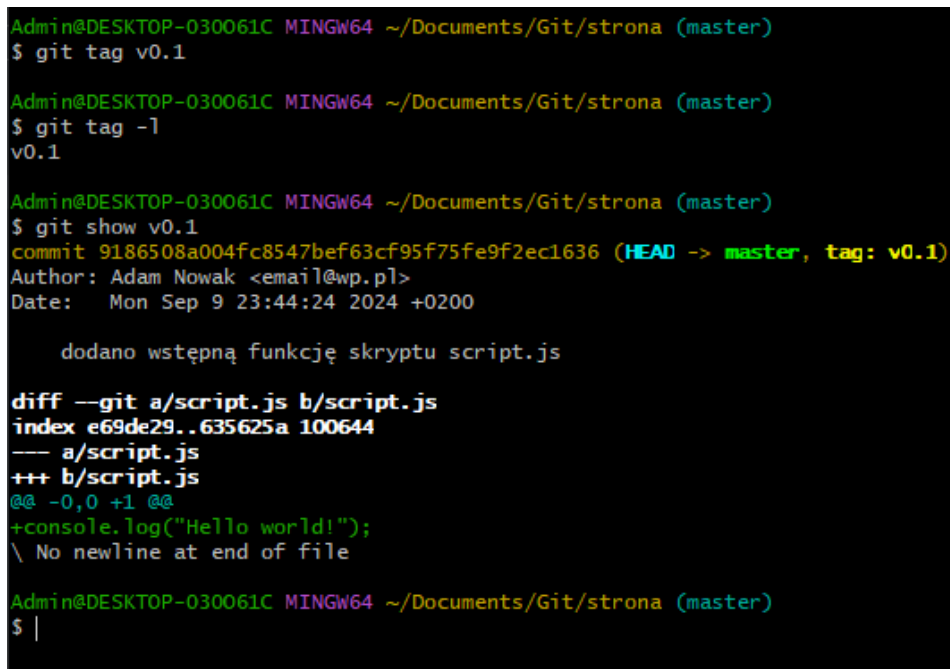
```
git tag -a [etykieta] -m [wiadomość]
```

Aby wyświetlić listę wszystkich tagów w repozytorium używamy polecenia:

```
git tag -l
```

Aby sprawdzić detale otagowanej migawki używamy komendy:

```
git show [etykieta]
```



```
Admin@DESKTOP-030061C MINGW64 ~/Documents/Git/strona (master)
$ git tag v0.1

Admin@DESKTOP-030061C MINGW64 ~/Documents/Git/strona (master)
$ git tag -l
v0.1

Admin@DESKTOP-030061C MINGW64 ~/Documents/Git/strona (master)
$ git show v0.1
commit 9186508a004fc8547bef63cf95f75fe9f2ec1636 (HEAD -> master, tag: v0.1)
Author: Adam Nowak <email@wp.pl>
Date: Mon Sep 9 23:44:24 2024 +0200

    dodano wstępną funkcję skryptu script.js

diff --git a/script.js b/script.js
index e69de29..635625a 100644
--- a/script.js
+++ b/script.js
@@ -0,0 +1 @@
+console.log("Hello world!");
\ No newline at end of file

Admin@DESKTOP-030061C MINGW64 ~/Documents/Git/strona (master)
$ |
```

Rysunek 2: Przykład lekkiego otagowania aktualnej migawki etykietą "v0.1", sprawdzenia listy tagów oraz wyświetlenia informacji o otagowanej migawce.

Możemy również szukać konkretnych tagów podając szukaną etykietę w następujący sposób:

```
git tag -l [szukana etykieta]
```

Przykładowo, jeżeli chcemy wyszukać wszystkie tagi z etykietami zaczynającymi się na v1.2 powinniśmy wpisać:

```
git tag -l v1.2*
```

Istnieje również możliwość otagowania jednej z poprzednich migawek, dodając do komendy "git tag" sumę kontrolną wybranej migawki w następujący sposób:

```
git tag [etykieta] [suma kontrolna]
```

```

Admin@DESKTOP-030061C MINGW64 ~/Documents/Git/strona (master)
$ git log
commit 8ce198d1ad96ae87f2e990abe03bfd3b1a35da08 (HEAD -> master)
Author: Adam Nowak <email@wp.pl>
Date: Tue Sep 10 01:04:23 2024 +0200

    Dodano zawartość style.css

commit c1258871270d571be2fd8d9fcea86174723d32a0
Author: Adam Nowak <email@wp.pl>
Date: Tue Sep 10 01:03:43 2024 +0200

    Dodano zawartość index.html

commit d924bf52d90e4dd9ad9670d93ae44bc717fc722d
Author: Adam Nowak <email@wp.pl>
Date: Tue Sep 10 00:20:02 2024 +0200

    Usunięto podstronę galerii galeria.html

commit 9186508a004fc8547bef63cf95f75fe9f2ec1636 (tag: v0.1)
Author: Adam Nowak <email@wp.pl>
Date: Mon Sep 9 23:44:24 2024 +0200

    dodano wstępną funkcję skryptu script.js

commit 8fd9682cf8e8442a9b0a89f98a75f4e12f6c3f69
Author: Adam Nowak <email@wp.pl>
Date: Mon Sep 9 23:42:52 2024 +0200

    Initial commit

Admin@DESKTOP-030061C MINGW64 ~/Documents/Git/strona (master)
$ git tag v0.1.1 c1258871270d571be2fd8d9fcea86174723d32a0

Admin@DESKTOP-030061C MINGW64 ~/Documents/Git/strona (master)
$ git tag -l
v0.1
v0.1.1

Admin@DESKTOP-030061C MINGW64 ~/Documents/Git/strona (master)
$ git show v0.1.1
commit c1258871270d571be2fd8d9fcea86174723d32a0 (tag: v0.1.1)
Author: Adam Nowak <email@wp.pl>
Date: Tue Sep 10 01:03:43 2024 +0200

    Dodano zawartość index.html

```

Rysunek 3: Przykład sprawdzenia historii repozytorium, otagowania wybranej migawki etykietą "v0.1.1" oraz wyświetlenia informacji o otagowanej migawce.

Jeżeli chcemy usunąć jakiś tag możemy dokonać tego używając komendy:

```
git tag -d [etykieta]
```