Git i platforma GitHub

1 Cel laboratoriów

Zapoznanie się z działaniem Git i platformy GitHub. Założenie konta i nauka współdzielenia źródeł. Git jest systemem kontroli wersji, którego znajomość jest często wymagana w firmach.

2 Słownik

- **SCM** – to akronim od Source Code Management, czyli dosłownie kontrola kodu (w języku polskim funkcjonuje określenie system kontroli wersji); jest to system który pozwala na archiwizowanie i śledzenie zmian w kodzie, dzięki czemu możemy cofać się w historii lub podejrzeć, kto był autorem konkretnej zmiany
- **Repozytorium** – ‚kontener' na określony zbiór kodu, najczęściej jeden projekt; repozytorium pozwala grupować kod i zmiany, dzięki czemu możemy przeglądać wszystkie zmiany wykonane w ramach jednego repozytorium, przyznawać uprawnienia do repozytoriów oraz pobierać / kopiować je
- **Commit (lub rewizja)** – jest to proces ‚wysłania' na repozytorium określonych zmian w kodzie – jeśli pobierasz kod z repozytorium, następnie dokonujesz modyfikacji i wysyłasz te zmiany z powrotem do repozytorium, proces ten nosi nazwę commitowania, a same zmiany wysłane razem nazywamy commitem lub rewizją
- **pull / push** – odpowiednio pobranie i wysłanie zmian (jednego lub wielu commitów) z/do innego repozytorium
- **diff** – (ang. różnica) – jest to różnica pomiędzy różnymi rewizjami – dzięki temu możemy zobaczyć, które fragmenty uległy zmianie oraz w jaki sposób; pozwala to także zoptymalizować transfer danych pomiędzy repozytoriami
- **fork** – kopia repozytorium; szczególnie popularne w przypadku projektów open-source, dzięki czemu możemy skopiować cały projekt i rozwijać go niezależnie (np. dopasowując do naszych potrzeb)
- **branch** – odgałęzienie, wersja wewnątrz repozytorium; branche pozwalają na prace wielu osobom równocześnie, bez ciągłego wchodzenia sobie w drogę i nadpisywania zmian – każdy może pracować na swoim branchu, dopiero po zakończeniu pracy łącząc zmiany z innymi i rozwiązując problemy

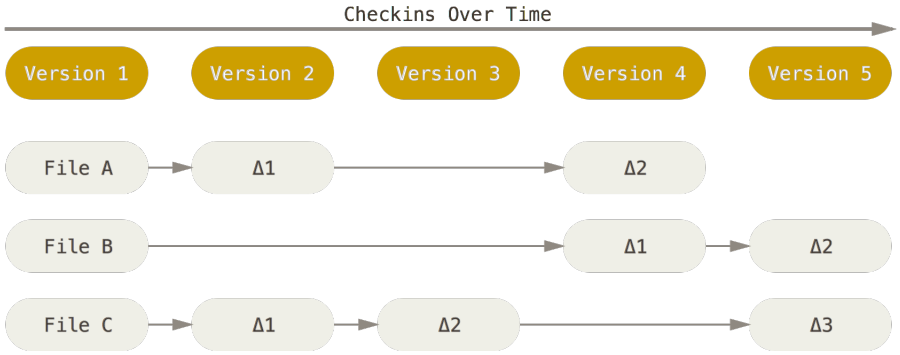
- **merge** – połączenie wielu zmian z różnych źródeł, które może skutkować niekompatybilnymi zmianami wymagającymi ręcznych modyfikacji; merge pozwala łączyć prace wykonywane w różnych obszarach, które mogą się zazębiać, w jedną całość w sposób kontrolowany i świadomy.[1]

2.1 Git

Git to system kontroli wersji, którego twórcą był Linus Torvalds. Git jak każdy system kontroli wersji zarządza i archiwizuje kolejne poprawki w projekcie. Jest głównie przeznaczony dla kodu, ale może być również używany do plików takich jak dokumenty Worda.

Niektórzy poprzednicy Git, tacy jak CVS i Subversion, posiadają centralne „repozytorium" wszystkich plików związanych z projektem. Gdy programista wprowadza zmiany, to robi to bezpośrednio w centralnym repozytorium. W rozproszonych systemach kontroli wersji, takich jak Git, jeśli chce się wprowadzić zmiany w projekcie, to można skopiować całość repozytorium w lokalną kopię. Można dokonać zmian właśnie w kopii lokalnej, i poprzez „check in" dokonać zmian w centralnym repozytorium. Taki system zachęca do tworzenia granularnych zmian, ponieważ nie wymusza się łączności z serwerem za każdym razem kiedy dokonuje się zmiany.

Różnice pomiędzy działaniem systemów kontroli wersji takich jak CVS, Subversion, Perforce, Bazaar wyjaśnia Rysunek 1 i 2. W Git za każdym razem, gdy wykona się zatwierdzenie zmian tzw. „commit" lub zapisze projekt, wówczas Git zapamiętuje stan (jako migawkę) wszystkich plików projektu. Jeżeli plik się nie zmienił, przechowuje się tylko link do pliku, który był w całości zapisany w chwili utworzenia lub kolejnych zmian.

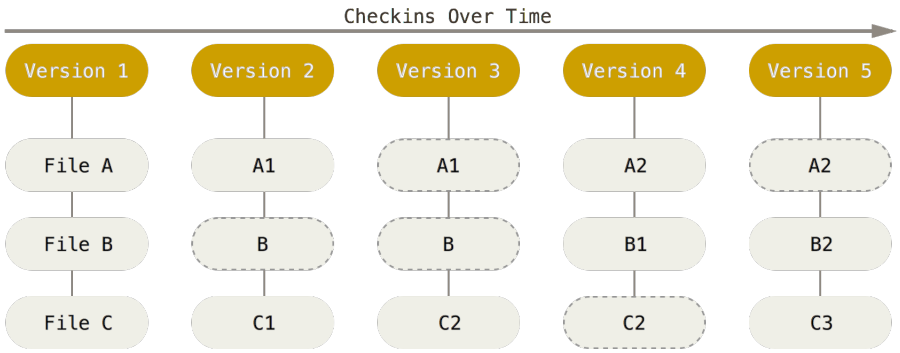


Rysunek 1: Kontrola wersji jako lista zmian w plikach źródłowych: źródło:http://git-scm.com/book/en/v2/Getting-Started-Git-Basics

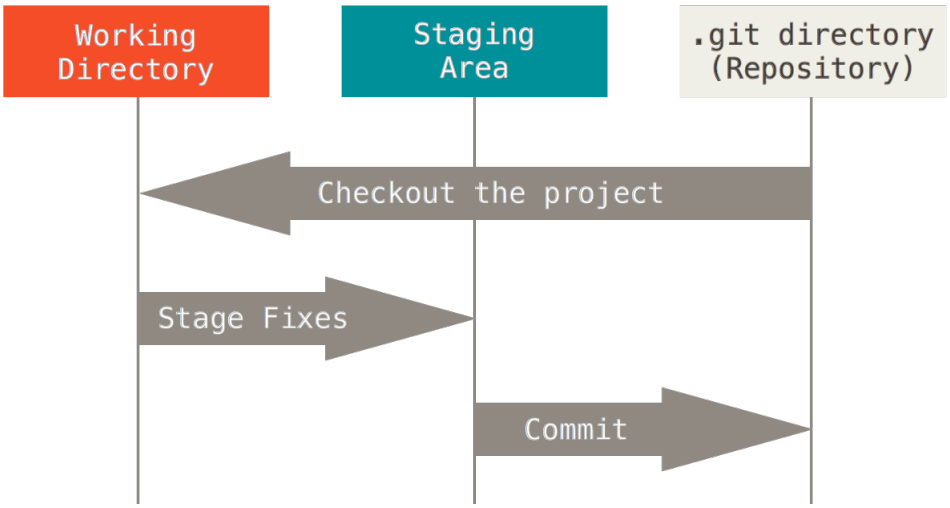
Główne cechy GIT to:

1. Prawie wszystkie operacje wykonywane są lokalnie — historia zmian jest przechowywana lokalnie jak i porównania plików z poprzednimi wersjami.

[1] `http://kobietydokodu.pl/niezbednik-juniora-git-kontrola-wersji/`



Rysunek 2: Kontrola wersji w Git: źródło:http://git-scm.com/book/en/v2/Getting-Started-Git-Basics



Rysunek 3: Trzy stany pliku. źródło:http://git-scm.com/book/en/v2/Getting-Started-Git-Basics

2. Integralność — każda operacja jest związana z generowaniem sumy kontrolnej (SHA-1) na pliku. Suma ta ma format 40 znakowego ciągu heksadecymalnego np. $24b9da6552252987aa493b52f8696cd6d3b00373$

3. Dodawanie — Git głównie dodaje elementy, usuwanie jest wieloetapowym zadaniem i ze względu na złożoność zabezpiecza przed niekontrolowaną utratą danych.

4. Trzy stany — pliki mogą się znajdować w jednym z trzech stanów: (1) zmodyfikowany, (2) staged (plik został zmodyfikowany i czeka by wpisać go do bazy zmian) i (3) commited (pliki zapisane w bazie zmian) (rysunek 3).

3 GitHub

„GitHub – hostingowy serwis internetowy przeznaczony dla projektów programistycznych wykorzystujących system kontroli wersji Git. Stworzony został przy wykorzystaniu frameworka Ruby on Rails i języka Erlang. Serwis działa od kwietnia 2008 roku . W kwietniu 2011 ogłoszono, iż GitHub obsługuje 2 miliony repozytoriów. Github udostępnia darmowy hosting programów open source oraz płatne prywatne repozytoria."[2]

Główne cele:

1. wymiana kodu i serwis publikowania kodu,
2. website społeczny dla programistów.

3.1 GitHub

GitHub jest usługą hostingową dla repozytorium Git i dodaje wiele własnych elementów. Git jest narzędziem obsługiwanym z linii komend, GitHub zapewnia graficzny interfejs webowy. Zapewnia również kontrolę dostępu i kilka możliwości współpracy, takich jak wiki i podstawowe narzędzia zarządzania zadaniami dla każdego projektu.

Flagową funkcjonalnością GitHub jest „forking" - kopiowanie repozytorium z konta jednego użytkownika do drugiego. Umożliwia to wykorzystanie projektu, do którego nie masz dostępu do zapisu ale możesz modyfikować go na swoim koncie. Jeśli dokonasz zmian, które chcesz udostępnić, możesz wysłać zgłoszenie zwane „pull request" do pierwotnego właściciela. Właściciel może następnie, za pomocą kliknięcia, scalić zmiany znalezione w przesłanym „repo" z oryginalnym „repo".

Dotychczas, jeżeli chciało się włączyć do projektu open source trzeba było ręcznie pobrać kod źródłowy projektu, wprowadź zmiany lokalnie, następnie utworzyć listę zmian zwanych „patch", a następnie wysłać mailem do opiekuna projektu. Opiekun musiał ocenić tę poprawkę, czasami wysłaną przez zupełnie nieznanego kodera, i zdecydować, czy do zmiany nadają się do scalenia.

W środowisku sieciowym jakim jest GitHub ta procedura wygląda inaczej. Kiedy koder złoży „pull request", opiekun projektu może zobaczyć jego profil, który obejmuje wkład w projekty zapisane w GitHub. Jeśli poprawka zostanie przyjęta, można uzyskać kredyt od opiekuna, i to widać w profilu. GitHub jest jak CV, które pomaga określić dorobek programisty. Im więcej ludzi i projektów na GitHub, tym lepszy obraz daje to opiekunom projektów. Poprawki mogą być publicznie dyskutowane.

Alternatywą dla GitHub jest bitbucket.org czy gitlab.com.

4 Zadania do wykonania

Celem zadań jest nauka wykorzystania Git i GitHub wraz z założeniem konta.

[2] http://pl.wikipedia.org/wiki/GitHub

4.1 Git - instalacja i konfiguracja

1. Przejdź do strony: `http://git-scm.com/downloads` lub `https://git-for-windows.github.io`. Drugi adres zawiera narzędzia okienkowe do obsługi Git.

2. Zainstaluj Git

3. Otwórz Git Bash, który jest CLI (Command Line Interface). Przetestować w nim polecenia: pwd, clear, ls, ls -l, ls -a, ls -al, cd .., cd, mkdir, touch, cp, rm, mv, echo, date.

4. Każde zatwierdzenie zmian „commit" będzie oznaczone nazwą użytkownika i adresem e-mail. Dlatego wykonaj następujące polecenia

   ```
   $git config --global  user.name ,,Your name here''
   $git config --global  user.email ,,Your email here''
   ```

   Wykonanie polecenia po raz kolejny doprowadzi do zmiany tych danych.

5. Sprawdź zmiany

   ```
   $git config --list
   ```

6. Aby wyjść z basha wpisz:

   ```
   $exit
   ```

4.2 GitHub - założenie konta i konfiguracja

1. Przejdź do strony: `https://github.com/`

2. Wprowadź: username, email i password i kliknij „Sign up for GitHub"

3. Uwaga: użyj tego samego adresu, który został użyty do konfiguracji Git

4. Na następnym ekranie kliknij „Free plan„ and kliknij „Finish sign up"

5. Na ekranie powitalnym wiele pomocnych linków (sprawdź)

6. Kliknij na swoje imię w prawym górnym rogu, by zobaczyć swój profil. Tu znajdują się informacje o aktywności na koncie, kim jesteś i nad czym pracujesz, powoli to miejsce stanie się Twoim portfolio. Można swój profil uzupełnić (wg uznania).

7. Poeksploruj portal ;-) znajdź kolegów i ciekawe projekty.

4.3 Lekcja - nauka poleceń i działania zdalnie

Wykorzystaj adres: `http://try.github.io`

4.4 GitHub tworzenie repozytorium

Git jest na lokalnym komputerze, GitHub na serwerze (współdzielenie danych oraz kopia zapasowa Twojej pracy).

Dwie metody tworzenia repozytorium

1. własne z plików
2. fork innego projektu

Metoda pierwsza:

1. Idź do strony: (`https://github.com/yourUserNameHere/`) i kliknij na „Create a new repo" w prawym górym rogu lub
2. Idź do strony `https://github.com/new` zaloguj się.
3. Nadaj nazwę i wpisz krótki opis
4. Wybierz „Public" (private jest dostępny tylko dla użytku komercyjnego lub edukacji)
5. Zaznacz box przy „Initialize this repository with a README"
6. Kliknij „Create repository"

Stworzenie lokalnej kopii:

1. Otwórz Git Bash
2. Wykonaj polecenia

```
$ mkdir ~/test-repo
$ cd ~/test-repo
$ git init
$ git remote add origin https://github.com/yourUserNameHere/test-repo.git
```

Metoda druga:

1. Znajdź repo na GitHub, które jest godne uwagi i kliknij na „Fork" (`https://help.github.com/articles/fork-a-repo`)
2. Lokalna kopię repozytorium (do katalogu bieżącego!) wykonuje się poleceniem

```
$ git clone https://github.com/yourUserNameHere/repoNameHere.git
```

5 Literatura obowiązkowa

1. `https://git-scm.com/book/pl/v1/Pierwsze-kroki-Podstawy-Git`