Wprowadzenie do Gita

Autor: Mateusz Krawczuk

0. Sprawdzenie, czy na komputerze jest git i pobranie tej instrukcji
    a) W terminalu wpisujemy *$ git clone https://github.com/mkrawczuk/git_101*
    b) *$ cd wprowadzenie_do_gita*
    c) otwieramy tę instrukcję

1. Czym jest git, jak i dlaczego powstał

2. Tworzenie konta na GitHub:
    a) Wchodzimy na https://github.com/ ,
    b) rejestrujemy nowe konto.

GitHub jest serwisem hostingowym przeznaczonym dla repozytoriów Gita. Alternatywa: Bitbucket

3. Konfiguracja Gita na komputerze;
    a) *$ git config --global user.name "NazwaUżytkownika"*,
    b) *$ git config --global user.email "adres@naszego.emaila*,

To są informacje kontaktowe, które dołączone będą do naszych commitów

4. Tworzenie własnego repozytorium:
    a) Wchodzimy na https://github.com/,
    b) W prawym górnym rogu, po prawej stronie od naszej nazwy użytkownika, wybieramy znak „+”->”Create new...”->”New repository”,
    c) wypełniamy pola „Repository name” i „Description”,
    d) wybieramy opcję „Add .gitignore”, szukamy „C++” i zaznaczamy,
    e) wybieramy „Create repository”.

5. Pobieranie repozytorium na swój komputer;
    a) Wpisujemy *$ git clone https://github.com/${NazwaUzytkownika}/${NazwaRepo}*,
        Uwaga: adres możemy skiopiować np. z paska adresu
    b) cieszymy się stworzeniem swojego repozytorium.

6. Praca na swoim repozytorium
    a) Kopiujemy zawartość katalogu git_101 do naszego repozytorium (cp ${skąd} ${dokąd}),
    b) $ git add inc,
    c) sprawdzamy *$ git status*,
    d) dodajemy poleceniem *add* jeszcze katalog src i plik CmakeLists.txt,
    e) commitujemy do zewnętrznego repo: $ *git commit*, wpisujemy tzw. commit message,
    Uwaga: commit message to krótka informacja o tym, co zostało zmienione w projekcie.
    f) wpisujemy $ *git push* – wrzucamy nasz projekt na zewnętrzne repo,
    g) wchodzimy na nasze repo – nieskończona radość.

7. Praca na czyimś repozytorium:
    a) Wchodzimy na repo drugiej pary w portalu GitHub – forkujemy repozytorium.
    Uwaga: w panelu repozytorium, na górze po prawej stronie znajduje się przyciskc „Fork”.
    b) Analogicznie jak w kroku 5 – pobieramy forka repozytorium innej grupy,
    Uwaga: klonujemy forka repo (na naszym koncie), a nie repo innej grupy (na ich koncie)!
    c) uzupełniamy kod w pliku src/calc.cpp,
    d) nalogicznie jak 6 b) i dalej,
    e) czekamy aż druga para skończy,
    f) wchodzimy na GitHub,

g) Zapoznajemy się z „Pull request”,
h) Nieskończona radość.

8. Branche – ochrona przed zepsuciem działającego kodu:
a) Wpisujemy *$ git branch*,
b) podziwiamy listę istniejących gałęzi,
c) *$ git checkout -b add*,
Tworzymy w ten sposób nowy branch o nazwie 'add' i przechodzimy do niego. Teraz możemy bezpiecznie zmieniać kod – mamy gwarancję, że zawsze możemy wrócić do działającej wersji projektu.
d) powtarzamy punkt 8 b),
e) staramy się zaimplementować nową metodę klasy Calculator – add(), która dodawać będzie dwa inty,
f) postępujemy analogicznie jak w 6 a)-e) – „zapiszemy” w ten sposób zmiany
g) *$ git checkout master*
h) sprawdzamy zawartość projektu i jest nam smutno, bo „zniknęła” gdzieś metoda add(),
i) *$ git checkout add*
j) ponownie oglądamy nasz projekt – „wróciła” metoda add(). Nieskończona radość.
k) jeżeli jesteśmy całkowicie pewni, że nasz nowy kod działa, możemy go dołączyć do gałęzi master:
l) *$ git checkout master,*
m) $ git merge add.

Znasz już techniki, które pozwalają na wydajną i bezpieczną pracę z Gitem. Umiesz wykorzystać podstawowe funkcje Gita do rozwijania projektów w pojedynkę i małej grupie. Mam nadzieję, że od teraz będziesz wykorzystywać zdobytą na tych zajęciach wedzę. Przekonasz się wtedy, jak stosowanie VCSa zwiększa wygodę pracy i współpracy.