Wstęp do współpracy z GIT-em.

Najważniejsze pojęcia:

commit - [komit] - punkt odniesienia do którego można przejść za pomocą komend

branch - [brancz] - gałąź kodu, zbiór uporządkowanych commitów, które charakteryzują pracę

master - [master] - główna gałąź kodu, kod źródłowy projektu

merge - [merdż] - proces łączenia dwóch gałęzi kodu w jedną

Podstawowe zasady:

1. Kod budujemy na własnym branchu, który mergujemy z masterem tylko gdy jest on stabilny.

2. Merge najpierw odbywamy między masterem a naszym branchem, a po rozwiązaniu konfliktów - naszym branchem a masterem.

3. Dobrym zwyczajem na rozpoczęcie pracy w danej sesji jest ściągnięcie danych z githuba, zmergowanie z naszym branchem, a później praca nad własnym kodem.

Podstawowe komendy GIT-a:

Jeśli korzystasz z konsoli GIT-a, to działania są wpisywane tekstowo. GIT posiada funkcję autouzupełniania podstawowych komend.

git clone <url/ssh>

ściąga cały projekt do lokalizacji w której znajduje się konsola

git branch <nazwa>

tworzy branch o podanej nazwie

git checkout <nazwa>

przechodzi do brancha o podanej nazwie

git add <nazwa pliku>

dodaje plik do następnego commita

git add .

dodaje wszystkie pliki poza zdefiniowanymi w pliku .gitingnore do następnego commita

git commit -m "<treść>"

dodaje commit ze zdefiniowanymi plikami; treść powinna jednoznacznie mówić, czego dotyczy commit

git reset

resetuje zmiany wprowadzone jako "git add <...>", powracając do pustego commita

git reset --hard

resetuje zmiany jak powyżej oraz resetuje treść plików (uwaga! występuje utrata dotychczasowej pracy)

git fetch

ładuje zmiany z githuba na dysk, ale nie wprowadza ich do lokalnego kodu; dotyczy wszystkich branchy zdefiniowanych na githubie; wprowadzenie zmian do kodu może odbyć się za pomocą komendy *git merge*

git pull

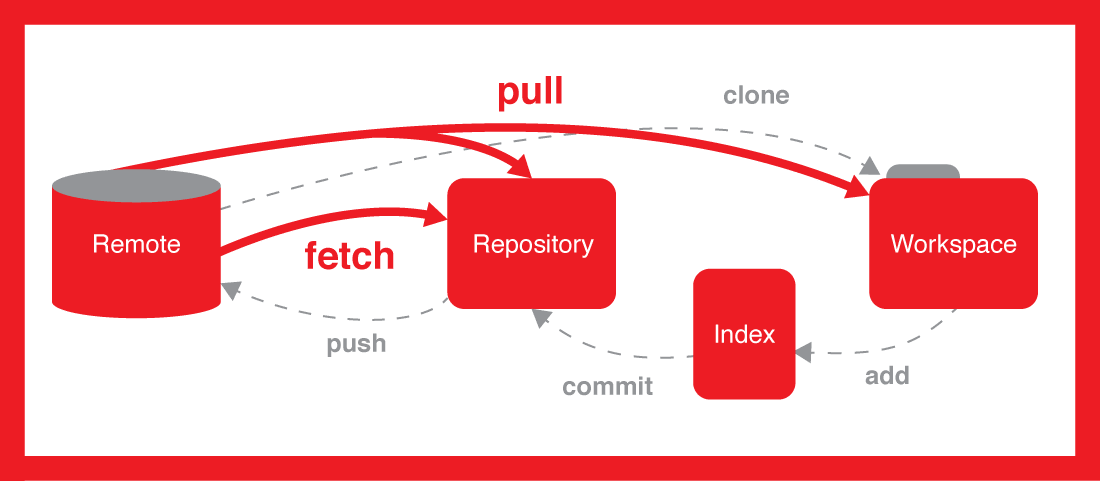
ładuje zmiany z githuba i wprowadza je do kodu; dotyczy aktywnego brancha, dlatego zalecane jest wykonywanie tej komendy przy aktywnym branchu *master*

git push

przenosi zmiany lokalne do githuba; dotyczy aktywnego brancha, dlatego zalecane jest wykonywanie tej komendy przy aktywnym branchu *master*

Trochę grafiki

Generalnie zmiany lokalne i zdalne dobrze przedstawia obraz:



Remote - github

Repository - folder .git na twoim dysku

Index - zmiany śledzone przez GITa

Workspace - kod źródłowy dostępny z pozycji edytora tekstu

pull - zaciąga zmiany zarówno do folderu .git jak i kodu źródłowego

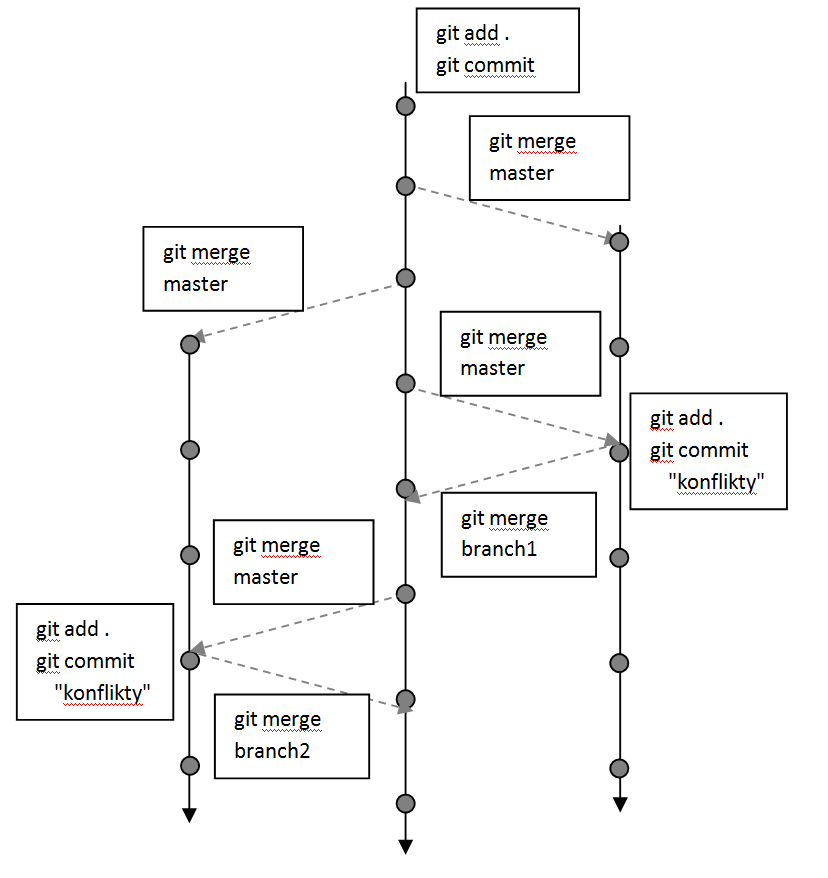
fetch - zaciąga zmiany tylko do folderu .git

add - dodaje zmiany kodu

commit - potwierdza zmiany kodu

push - przesyła potwierdzenia zmian kodu do githuba

To, jak chciałbym, żebyśmy pracowali na branchach przekazuje poniższa grafika:



Kropki to poszczególne commity. Gałąź środkowa to *master* i powinna być ona jak najbardziej stabilna (bez niepotrzebnych błędów). Gałęzie po bokach to gałęzie dwóch osób pracujących nad różnymi funkcjonalnościami. Pierwsza z nich pobiera zmiany z *mastera* i pracuje nad nimi. W tym samym czasie druga osoba również pobiera zmiany i pracuje w swoim zakresie. W międzyczasie do głównej gałęzi mogą dotrzeć inne zmiany (stąd commity pomiędzy zaciągnięciami). Przed mergem zmian pierwszej osoby, pobiera ona ostatnią wersję *mastera* i merguje ją z własną gałęzią. Rozwiązuje też ewentualne konflikty (git add, git commit). Następnie przesyła swoje zmiany do gałęzi głównej. Gdy druga osoba chce złączyć zmiany z *masterem* wykonuje te same czynności, czyli merguje gałąź z *masterem* i po rozwiązaniu konfliktów - merguje *master* ze swoją gałęzią.

Przykłady kolejnych poleceń

(pracujemy na gałęzi "branch1")

**1. Rozpoczęcie pracy**

git checkout master

git pull

(komunikat "fast-forward" albo "up-to-date")

git checkout branch1

git merge master

(może być komunikat o błędach, wtedy rozwiązanie ich i: )

git add .

git commit -m "rozwiązanie konfliktów"

**2. praca nad funkcjonalnością**

(po inicjalizacji z pkt 1 można rozpocząć pracę; trochę kodu)

git add .

git commit -m "szkielet nowej funkcjonalności"

(trochę kolejnego kodu)

git add .

git commit -m "działający interfejs"

(znowu coś nowego)

git add .

git commit -m "działający backend"

(dodanie czegoś)

git add .

git commit -m "połączenie wszystkiego w całość"

(znaleziono błąd i poprawiono go)

git add .

git commit -m "bugfix: niepotrzebny przecinek"

**3. mergowanie z masterem**

(kod wydaje się być stabilny i przechodzi testy)

git checkout master

git pull

git checkout branch1

git merge master

(rozwiązanie konfliktów)

git add .

git commit -m "rozwiązanie konfliktów"

git checkout master

git merge branch1

git push