Poradnik obsługi Atollic TrueSTUDIO w połączeniu z gitem (github).

# 1. Wstęp

Zakładamy, że do zalet używania systemów kontroli wersji nie trzeba nikogo przekonywać. W poradniczku skupimy się na obsłudze (wygodnym i świadomym poruszaniu się w środowisku) gita.

Git bash to narzędzie konsolowe zapewniające pełnię funkcjonalności związanych z obsługą systemów git. Z mojego punktu widzenia konsola to najlepsze rozwiązanie, gdyż widać dokładnie co się dzieje. Skupię się na git bashu.

Alternatywą jest używanie wbudowanej i preinstalowanej wtyczki do Atollica. Pozwala to bezpośrednio z IDE wykorzystać zalety gita.

# 2. Clone projektu na komputer i ustawienie pierwszego zdalnego dostępu do repo (tzw. Remote)

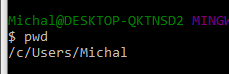
Na tym etapie zakładamy, że na github.com jest już założone repozytorium, do którego chcemy się łączyć na komputerze.

Celem działania jest podłączenie nie wykorzystując strony github.com, a wyłącznie synchronizując swoje pliki na komputerze lokalnym z serwerem github.

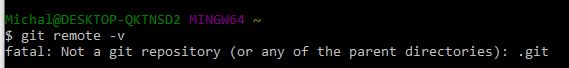
1) na stronie github.com zforkuj repo (chyba, że to Twoje, wtedy nie☺)

2) uruchom git basha

3) domyślnie powinieneś być w ścieżce C:\Users\Username (sprawdź poleceniem pwd)



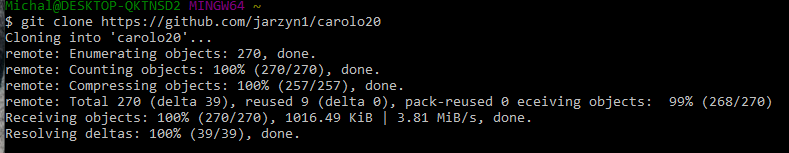
4) sprawdź jakie masz remote w tej ścieżce



Tutaj powinieneś nie mieć skonfigurowanych żadnych gitów, bo to ścieżka główna, od której dopiero będą odchodzić podfoldery z projektami!

5) clone projektu

* git clone <https://github.com/your-login/Carolo20>



W folderze C:/Users/Michal/Carolo20 został skopiowany projekt z https://github./com/Jarzyn1/Carolo20

6) przejście do folderu

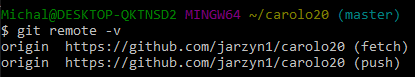
cd Carolo20

Sprawdzenie poprawności poprzez komendę pwd



7) sprawdzenie remote

Git remote –v

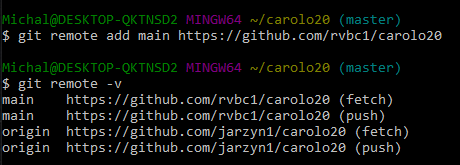


Na tym etapie masz jeden remote wskazujący na Twój github

8) Igor konfiguruje dodatkowo remote wskazujący na github Marka, Marek nie konfiguruje drugiego remote’a

git remote add main <https://github.com/rvbc1/>Carolo20

Teraz masz skonfigurowane 2 remote repos.



Ściągnięcie zmian online odbywa się poprzez:

**Git pull nazwa\_remote *nazwa\_brancha***

* Git pull main master
* Git pull origin master
* Git pull origin moj\_super\_kod\_dev\_niesprawdzony

# 3. Zmiany kodu lokalnie. Zaakceptowanie zmian

Commit&push

* git commit -a -m "treść commita"
* git push origin master

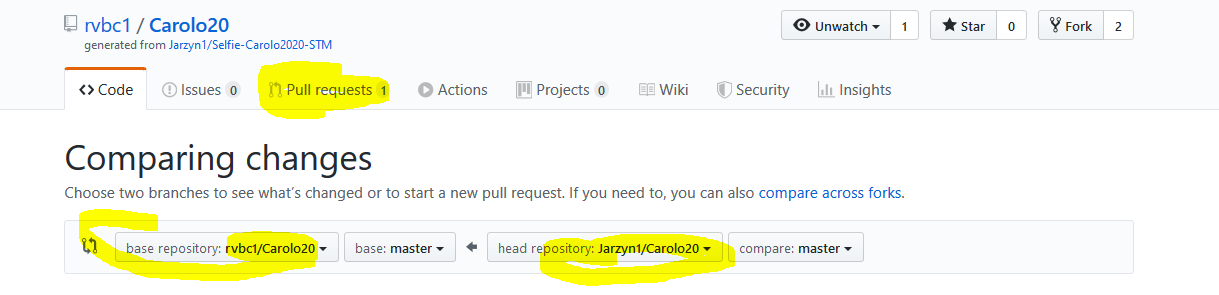
Kolejne poprawki wypuszczamy w commitach, opatrzonych odpowiedną, znaczącą nazwą.

Przed pushem warto sprawdzić, czy wszystkie pliki z commita uznawane przez working tree

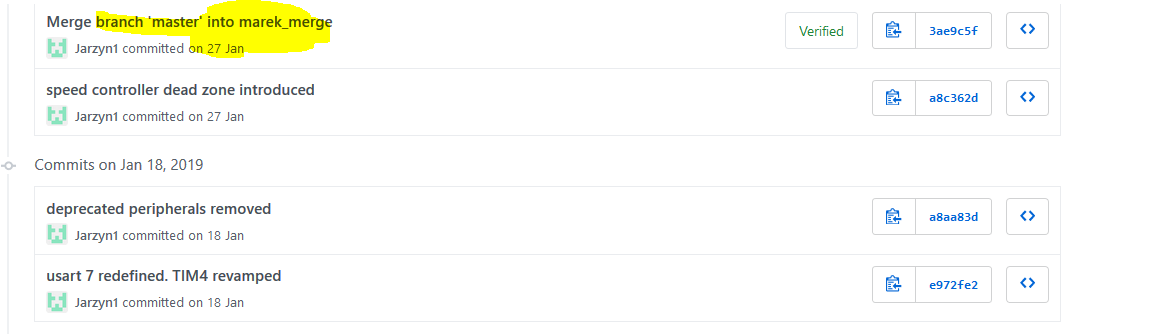
* Git status – sprawdzenie
* Git add \* - ewentualne dodanie wszystkich plików

Wypchnięcie zmian lokalnych odbywa się poprzez komendę **git push *nazwa\_remote nazwa\_brancha.*** W tym momencie zmiany pojawią się na naszym repo na github.com. Zawsze pushujemy na swoje repo, dopiero potem tworzymy pull request na repo główne.

Tworzenie nowego pull requesta odbywa się przez stronę. Pamiętaj o odpowiednim komentarzu, trafnych i rzetelnych opisach co zostało zmienione.



Po zaakceptowaniu pull requesta główny target przyjmuje historię commitów (zmian).



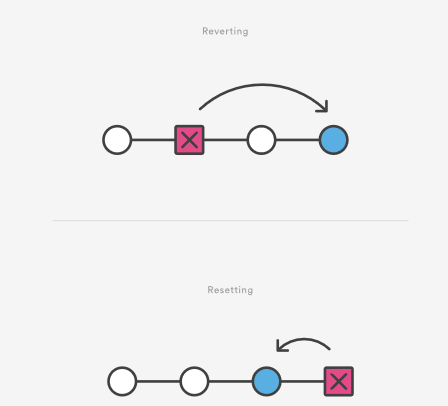
# 4. Cofanie zmian, gałęzie, oglądanie starych commitów

Do cofania zmian mamy dwa główne polecenia.

Git revert – polegający na cofnięciu efektów jednego, konkretnego, wybranego commita

Git reset – resetujący drzewo commitów wstecz, do konkretnego, wybranego commita

Najważniejsza różnica polega na tym, że git revert nie kasuje drzewa gita, a jedynie dodaje nowy commit, odwracający działanie wskazanego.



Najmocniejszym i najbardziej drastycznym w skutkach jest **git reset –hard *commit***

Aby obejrzeć stary commit używamy **git checkout commit**

Aby wrócić do aktualnej wersji **git checkout master.**

Tworzenie dodatkowych branchy to sposób na bezpieczne odseparowanie wersji przetestowanej i wgranej od developmentu.

**Git checkout –b *nazwa\_brancha*** – powoduje utworzenie i przejscie do nowego brancha z commita na którym obecnie jesteśmy

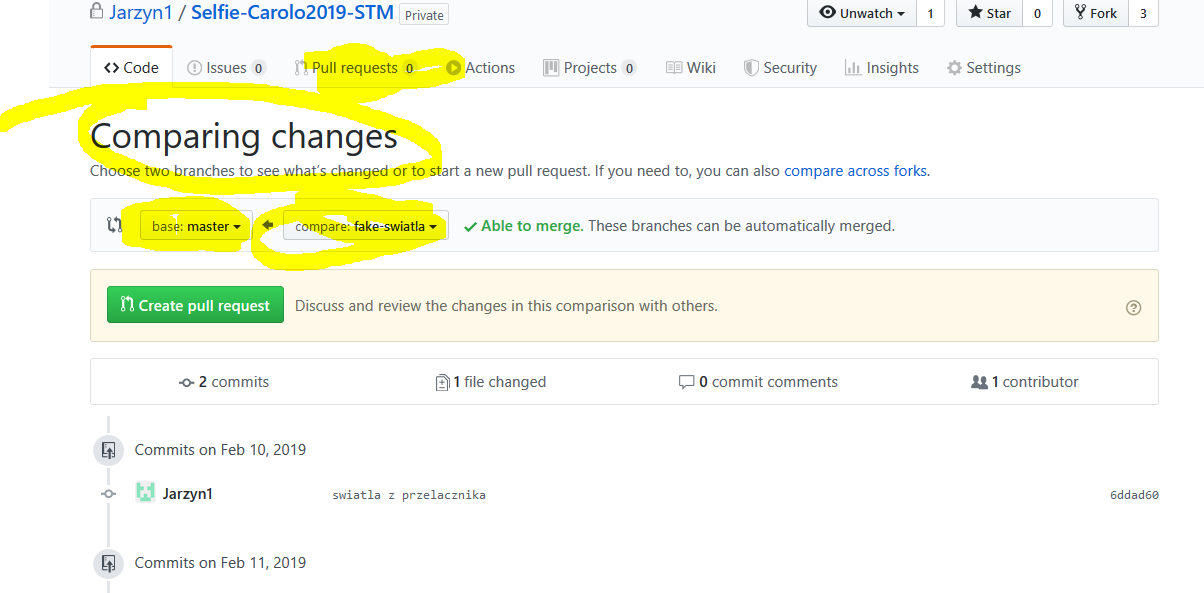
Po utworzeniu brancha możemy tradycyjnie robić commity i pushować zmiany online. Uwaga, przy pushu trzeba świadomie wybrać nazwę brancha **git push origin *nazwa\_brancha***.

Zmiany można scalić na dwa sposoby:

a) komendami git basha

* git checkout master
* git merge dev

b) przez pull request na stronie github.com. Pull requesty robi się w tym samym miejscu, ale wybiera swoje repo i dwa różne swoje branche.



Usunięcie niepotrzebnego brancha

**Git branch –d *nazwa\_brancha***

# 5. Pozostałe przydatne komendy

Git remote rm – usunięcie remote