**Git – System Kontroli Wersji (Ściąga)**

**Działanie na plikach:**

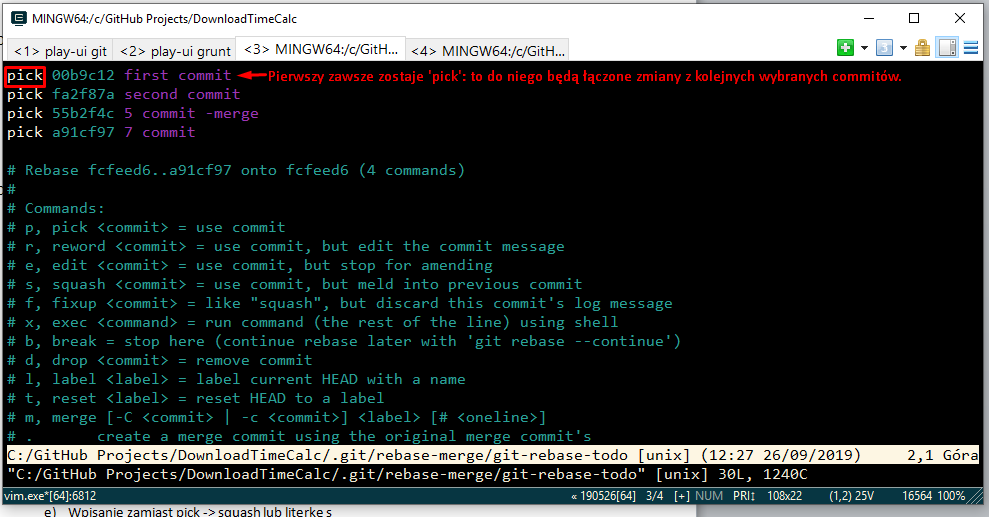
1. update-index --assume-unchanged <file> - ukrycie plików z poczekalni (git status)
2. update-index –no-assume-unchanged <file> - pokazanie ukrytych plików w poczekalni (git status)
3. !git ls-files -v | grep "^[[:lower:]]" – pokazanie listy ukrytych plików w poczekalni (git status)

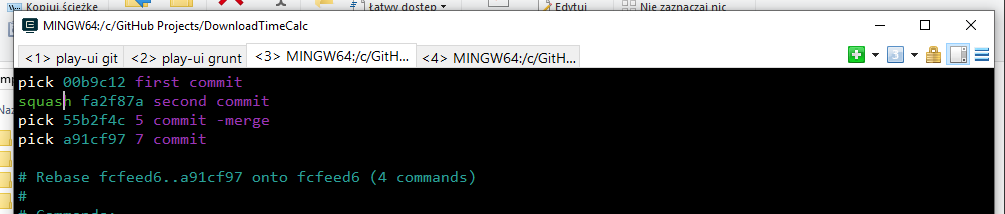
**Działanie na commit’ach:**

1. Git commit –m „Title” –m „Description…………” – dodanie szczegółów do commita.
2. Git commit: otwiera się vim i tam uzupełnianie commita Pierwsza linia title, druga pusta, trzecia szczegóły.
3. Git commit –amend: otwiera się vim do edytowania nazwy ostatniego commita.

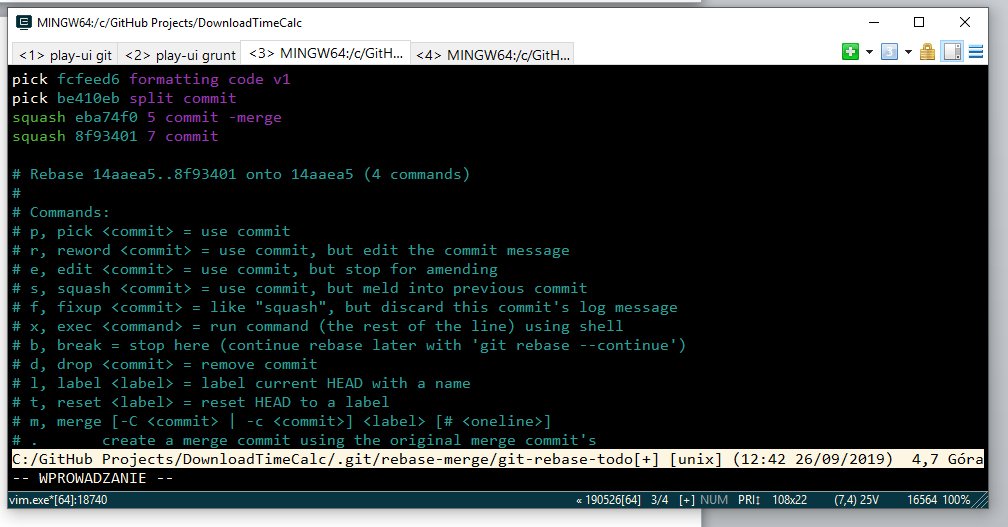
**Działanie na commitach w VIM’ie:**

1. Usunięcie danego commita:  
   a) git rebase –i HEAD~n (gdzie n liczba ostatnich commitów)
2. Wciśnięcie literki ‘i’ albo ‘a’, aby zacząć wpisywać
3. Najechanie na interesujący nas commit
4. Wpisanie zamiast pick -> drop lub kliknąć dwa razy literke ‘d’
5. !wq – wyjście i zapisanie zmian
6. Git push –f : wypchnięcie zmian z --forcem.
7. Połączenie dwóch commitów:
8. Użycie –squash: łączy commity i można edytować opis commita do którego będą te wszystkie zmiany zapisane.
9. git rebase –i HEAD~n (gdzie n liczba ostatnich commitów)
10. Wciśnięcie literki ‘i’ albo ‘a’, aby zacząć wpisywać
11. Najechanie na interesujący nas commit
12. Wpisanie zamiast pick -> squash lub literkę s
13. Pierwszy commit zawsze zostaje z pick i to do niego będą łączone zmiany.





1. !wq – wyjście i zapisanie zmian
2. Otworzy się ponownie VIM, aby wprowadzić edycję szczegółów commita itd.
3. !wq – wyjście i zapisanie zmian
4. Połaczenie kliku commitów:
5. Dalej analogicznie .



1. Zamiana miejscami commitów:
2. git rebase –i HEAD~n (gdzie n liczba ostatnich commitów)
3. Najechanie na interesujący nas commit
4. Kliknięcie dwa razy ‘d’ – spowoduje usuniecie commita
5. Przejście wyżej/niżej tam gdzie chcemy tego commita i klikamy ‘p’
6. !wq – wyjście i zapisanie zmian
7. Git push –f : wypchnięcie zmian z --forcem.
8. Usuwanie niechcianych plików z commita:
9. git reset --soft HEAD~n (gdzie n liczba ostatnich commitów) (ostatni commit to n = 1)
10. git status – pojawią się wszystkie pliki z tego commita w postaci git add
11. git reset HEAD <plik\_ktory\_chcesz\_usunac\_z\_commita>
12. git commit –m „nazwa\_commita” – edytowany commit zostanie nadpisany (można powiedzieć że już go nie ma, można wpisać taką samą nazwę jak edytowany commit)
13. Git push –f : wypchnięcie zmian z --forcem.

**Działanie na branch’ach:**

1. Git branch --a: wyświetlenie wszystkich branchy
2. Zmiana nazwy brancha
3. **Lokalnie:** git branch -m new-name.
4. **Na originie:** git branch -m new-name.
5. git push origin :old-name new-name
6. git push origin -u new-name
7. Usunięcie brancha:
8. **Lokalnie:** git branch -d branch\_name
9. **Na originie:** git push <remote\_name> --delete <branch\_name>

Example: git push origin --delete testowanie-vima

**Działanie na remote’ach:**

1. Git remote –v: wyświetlenie wszystkich remote’ów
2. Dodanie nowego remote’a:  
   a) git remote add <name> <https-url>

b) git fetch <name of remote>

3) Podgląd zmian w remotach: gitk –all

**Rebase jednego brancha na drugi:**

Jeżeli chcę na gałąź ‘master’ wrzucić zmiany z gałęzi przykładowo: ’ jira-621’:

1. Wchodzę na jira-621 (git checkout jira-621)
2. Robie git rebase master – (wtedy jira-621 leci na górę mastera)
3. Sprawdzam czy wszystko jest tak jak powinno.
4. Resetuje gałąź master do tej gałęzi.