GIT

systemy kontroli wersji (2/3)

Cześć!

Jestem Michał Ignaciuk

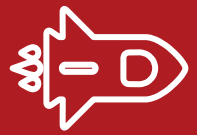
Java Web Applications Developer

michal.ignaciuk@gmail.com

Czym będziemy się zajmować?

- Merge - zasada działania i praktyka,
- Konflikty - skąd się biorą i jak je rozwiązywać,
- Rebase,
- git-flow,
- Obsługa GIT w IntelliJ IDEA,
- Usługi hostujące GIT - Github i Bitbucket,
- Pull request.

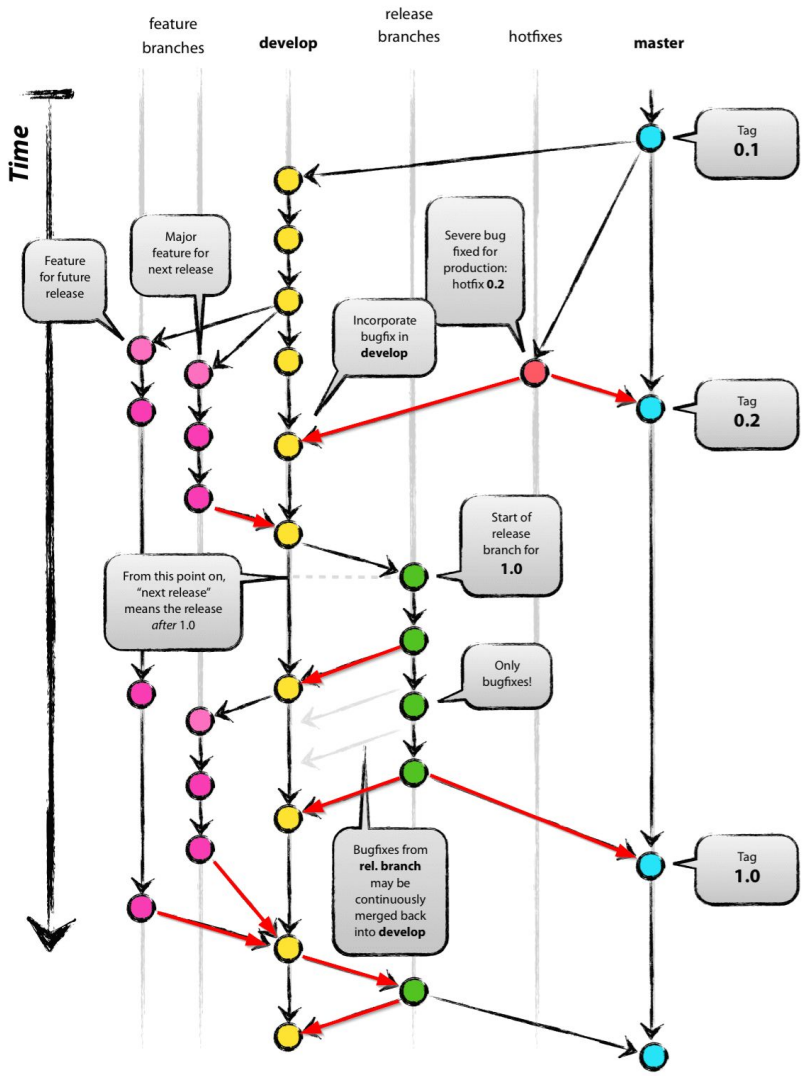
Merge



Zasada działania i praktyka

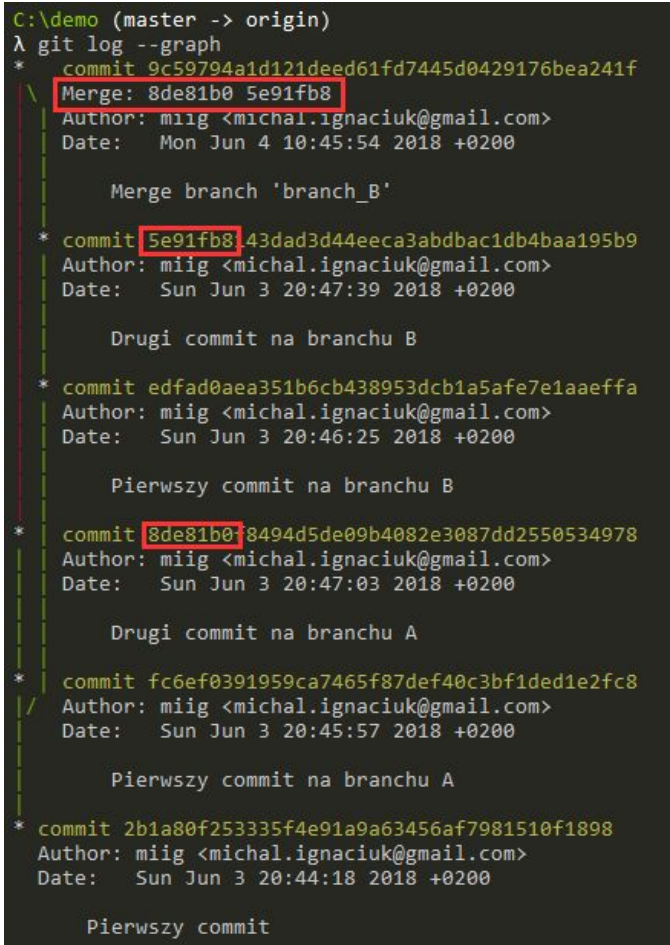
Merge
co i po co mergeujemy

- Mergeujemy jeden branch z drugim,
- Aby udostępnić swoje zmiany w kolejnej wersji aplikacji,
- Aby branch ja którym pracujemy był aktualny.



Merge
jak mergeujemy

- Mergeujemy poleceniem:
  - >> git merge branch_A
- Musimy być na branchu **do którego** chcemy zmergeować zmiany,
- Merge brancha skutkuje powstaniem merge-commita,
  - Zauważ, że merge commit ma dwóch "rodziców",



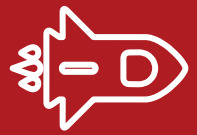
Merge
podstawowe operacja

- Aby utworzyć nowy branch:
  - >> git branch {nazwa}
- Aby przełączyć się na inny branch:
  - >> git checkout {nazwa}
- Aby zmergeować branch do aktualnego brancha:
  - >> git merge {nazwa}
- Aby zobaczyć historię wszystkich branchy:
  - >> git log --branches
- Aby zobaczyć historię w postaci grafu:
  - >> git log --graph

Merge
zadanie

- Stwórz lokalne repozytorium,
- W branchu master stwórz i zacommituj plik README.MD o dowolnej treści,
  - Pamiętaj aby treść commita miała jakiś sens ;)
- Stwórz nowe branche o nazwach branch_A i branch_B,
- Przełącz się na branch_A, dodaj nową linię o dowolnej treści do pliku README.MD i zacommituj zmiany - powtórz jeszcze raz z innymi zmianami,
- Przełącz się na branch_B, dodaj nową linię o dowolnej treści do pliku README.MD i zacommituj zmiany - powtórz jeszcze raz z innymi zmianami,
- Przełącz się na branch master i dodaj tam nowy plik index.html o dowolnej treści i zacommituj zmiany,
- **Skopiuj katalog .git do nowej lokalizacji (przyda się w kolejnym ćwiczeniu),**
- Zmergeuj branch branch_A do brancha master,
- Zobacz historię zmian na branchu master i zawartość pliku README.MD,
- Spróbuj zmergeować branch branch_B do brancha master...

Konflikty



Skąd się biorą i jak je rozwiązywać

Konflikty
skąd się biorą

- Jeżeli ta sama część tego samego pliku została zmieniona w inny sposób w różnych branchach które mergeujemy, GIT nie będzie w stanie połączyć tych zmian
  - To dobrze!

- Konflikty trzeba rozwiązać ręcznie
  - Mogą nam w tym pomóc specjalne narzędzia

Konflikty
jak wygląda konflikt

- Pliki z konfliktami możemy odnaleźć poleceniem:
  - >> git status

```
C:\demo (master -> origin)
λ git status
On branch master
You have unmerged paths.
  (fix conflicts and run "git commit")
  (use "git merge --abort" to abort the merge)

Unmerged paths:
  (use "git add <file>..." to mark resolution)

        both modified:   README.MD

no changes added to commit (use "git add" and/or "git commit -a")
```

```
Hello world!

Tu bedziemy dopisywac zmiany.

<<<<<<< HEAD
Pierwsza zmiana na branchu A.

Druga zmiana na branchu A.
=======
Pierwsza zmiana na branchu B.

Druga zmiana na branchu B.
>>>>>>> branch_B
```

- GIT zapisuje w pliku obie wersje konflitujących ze sobą zmian,
- Każda wersja opisana jest nazwą brancha albo commita.

Konflikty
jak go rozwiązać

- Najprościej (nie zawsze najłatwiej) - edytując plik w edytorze tekstowym,
- Można spróbować:
  - >> git mergetool (uwaga na vimdiff;)
- Zazwyczaj robimy to w swoim IDE (np. IntelliJ IDEA),
- Po rozwiązaniu konfliktu zmiany musimy zacommitować (!).

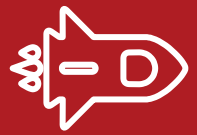
Konflikty
podstawowe operacja

- Aby sprawdzić jakie pliki mają konflikty:
  - >> git status
- Aby rozwiązać konflikt:
  - Uruchom swoje ulubione narzędzie :)
- Aby potwierdzić rozwiązanie konfliktu na pliku:
  - >> git add {nazwa.pliku}
- Aby zacommitować zmiany:
  - >> git commit (bez -m)
- Aby zobaczyć efekty:
  - >> git log --graph

Konflikty
zadanie

- ... próbowaliśmy zmergeować branch branch_B do brancha master,
- Sprawdź w jakim pliku jest konflikt,
- Wyedytuj plik w edytorze tekstowym (np. nano) tak aby rozwiązać konflikt,
- Zacommituj zmiany i sprawdź historię,
- Czy merge commit jest widoczny?

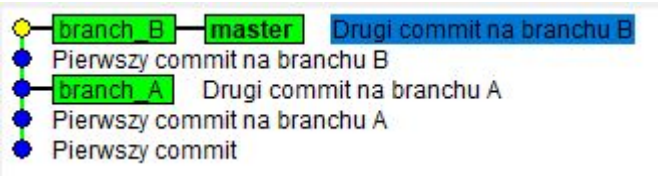
Rebase

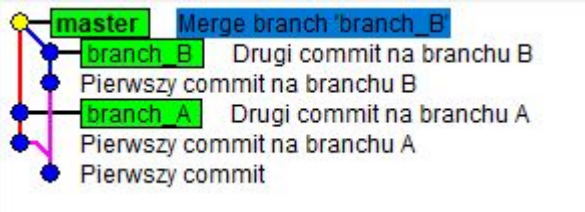


Alternatywa dla mergea

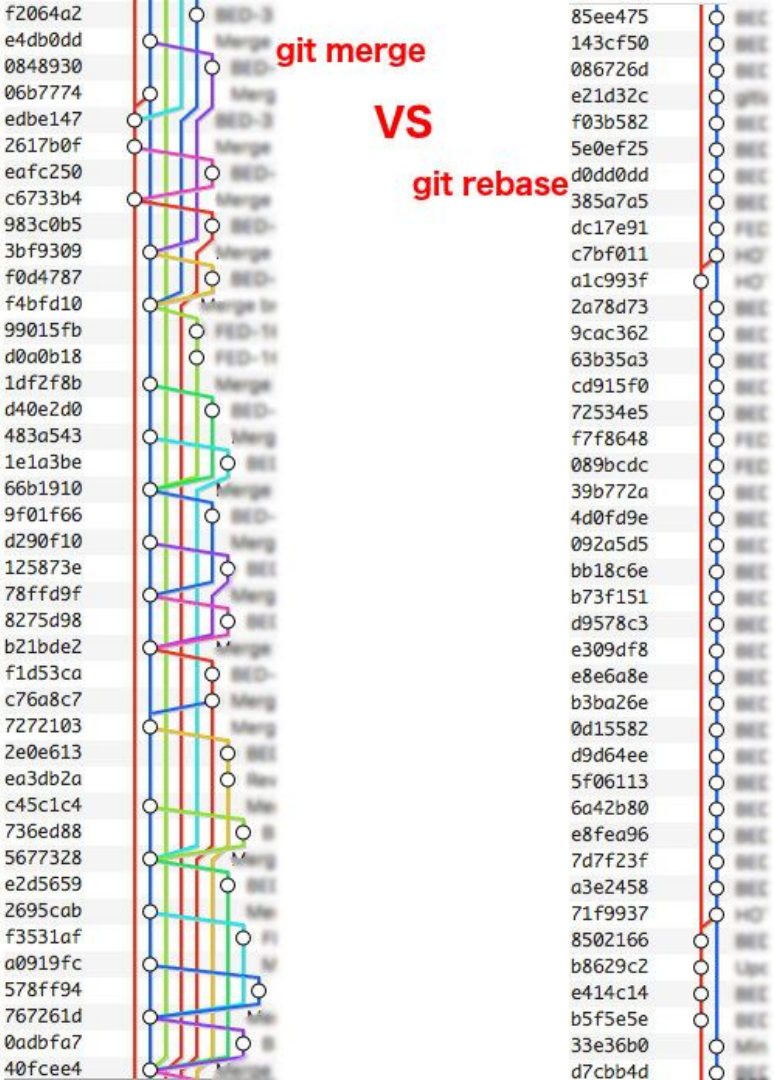
Rebase
czym różni się od merge'a

- Rebase przenosi zmiany z jednego brancha na drugi,
- Nie tworzy merge commit'a - dodaje zmiany z jednago brancha na koniec innego,
- Tworzy bardziej liniową historię zmian na branchu,
- Zmienia historię brancha:
  - **Nie powinien** być stosowany na publicznych branchach (!),
- Jako że zmiany są "nakładane" jedna po drugiej - może powodować więcej konfliktów.





Rebase
czym różni się od merge'a



źródło: https://www.flickr.com/photos/mariocarrion/31876330261

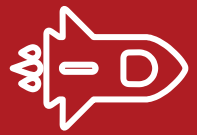
Rebase
podstawowe operacja

- Aby zrebase'ować zmiany z innego brancha do aktualnego brancha:
  - >> git rebase {nazwa}
- Aby kontynuować rebase po rozwiązaniu konfliktu:
  - >> git rebase --continue

Rebase
zadanie

- Przełącz się na skopiowane wcześniej repozytorium,
  - Jeżeli trzeba przełącz się na branch master i użyj >> git reset --hard
- Zmergeuj branch branch_A do brancha master,
- Zobacz historię zmian na branchu master i zawartość pliku README.MD,
- Przełącz się na branch branch_B i zrebase'uj zmiany z brancha master,
- Rozwiąż konflikty i kontynuuj rebase aż do skutku,
- Zacommituj zmiany,
- Zobacz historię zmian na branchu master i zawartość pliku README.MD,
- Czy widzisz różnicę w porównaniu do merge'a?

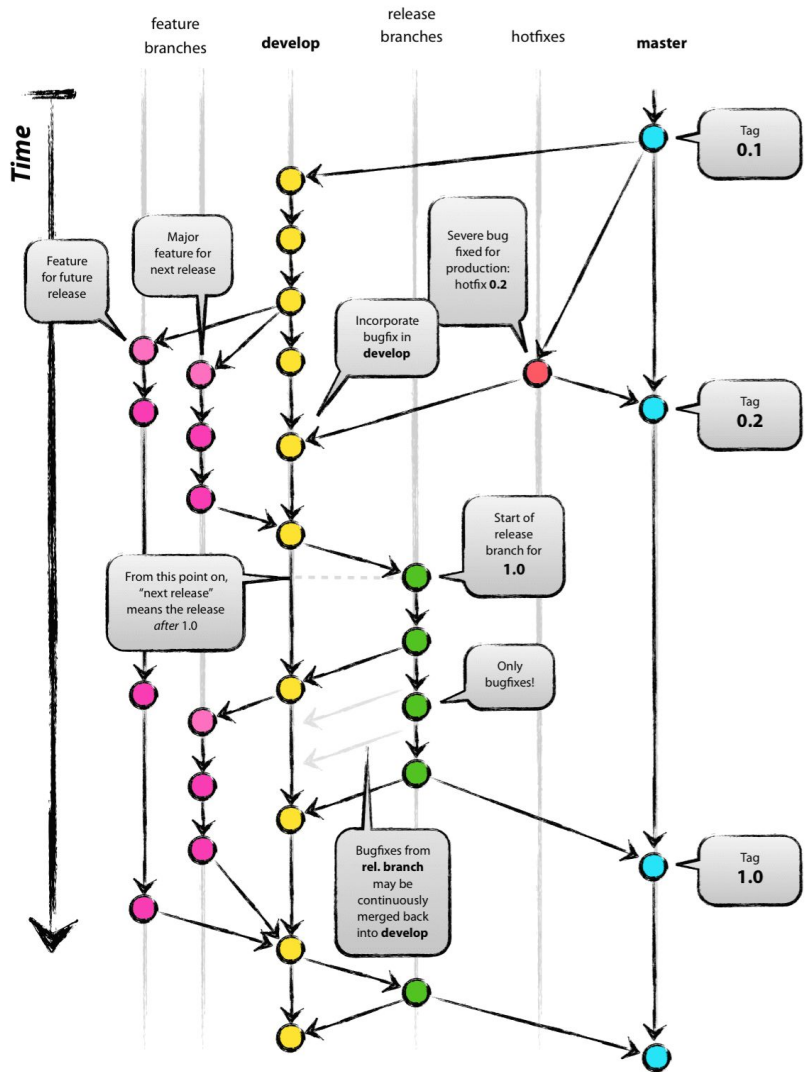
Git flow



Podsumowanie

Git flow
podsumowanie

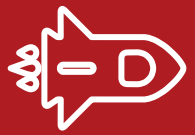
- Pracujemy na feature branchach,
  - Jedno zadanie/funkcja - jeden branch,
- Branche odpowiednio nazywamy, np.:
  - feature/FZ-54_some_feature
  - hotfix/FZ-54_some_feature
- Commity odpowiednio komentujemy, np.:
  - FZ-54 new awesome feature



Git flow
dobre praktyki

- Commituj szybko i często,
- Każdy commit reprezentuje zamkniętą całość (i przechodzi testy!),
- Każdy branch reprezentuje jeden feature,
- Wiadomości w commitach powinny być zwięzłe, konkretne i zrozumiałe,
- Należy często aktualizować swój feature branch z developa,
- Należy często przesyłać swoje zmiany na serwer (push),
- Merge zawsze poprzedza pull request.

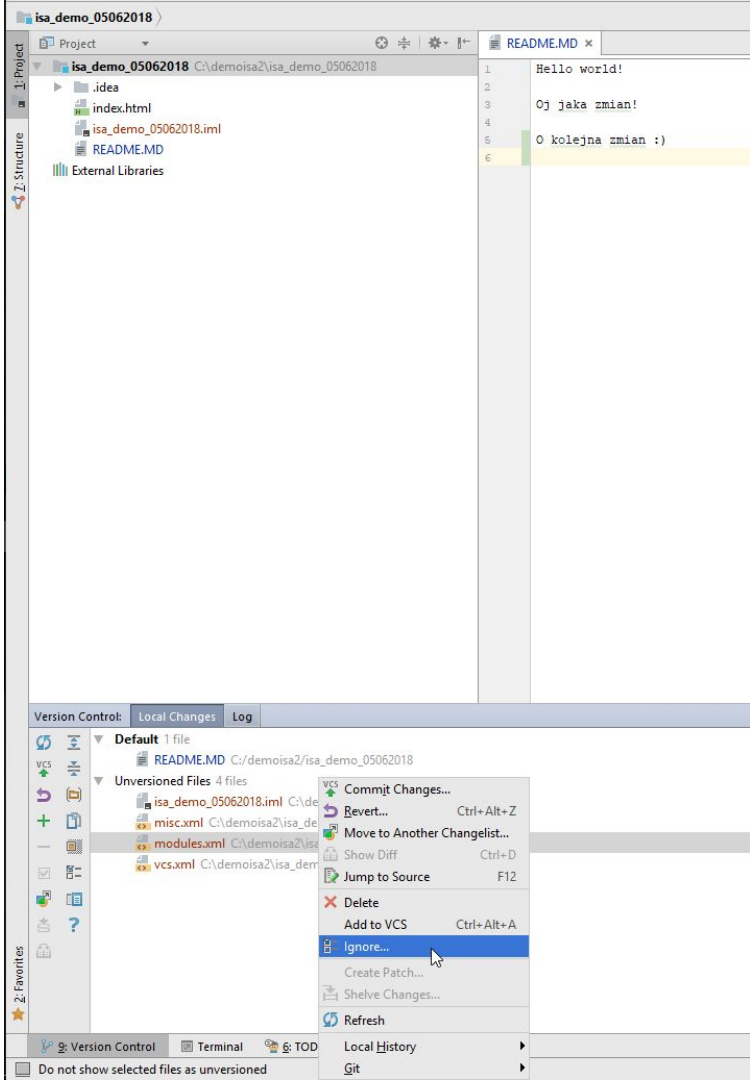
GIT w IntelliJ IDEA



Podstawowy workflow

GIT w IntelliJ IDEA
demo

- Utworzenie nowego projektu i wyciągnięcie kodu z github (albo lokalnego repozytorium),
- Dokonanie zmian w pliku,
- Dodanie nowego pliku,
- Dodanie plików do .gitignore,
- Stworzenie brancha,
- Przeglądanie historii repozytorium,
- Commit i push.



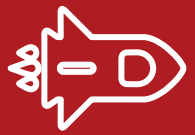
GIT w IntelliJ IDEA
zadanie 1/2

- Stwórz nowy projekt w IntelliJ IDEA (z istniejącego lokalnego repozytorium albo z github),
- Dodaj pliki których nie chcesz wersjonować do .gitignore,
- Zmień jakiś plik, dodaj nowy, zacommituj i zrób push'a,
- Zobacz historię repozytorium i pojedynczego pliku.

GIT w IntelliJ IDEA
zadanie 2/2

- Stwórz nowe branche o nazwach branch_A i branch_B,
- Przełącz się na branch_A, dodaj nową linię o dowolnej treści do pliku README.MD i zacommituj zmiany - powtórz jeszcze raz z innymi zmianami,
- Przełącz się na branch_B, dodaj nową linię o dowolnej treści do pliku README.MD i zacommituj zmiany - powtórz jeszcze raz z innymi zmianami,
- Przełącz się na branch master i dodaj tam nowy plik index.html o dowolnej treści i zacommituj zmiany,
- Zmergeuj branch branch_A do brancha master,
- Zobacz historię zmian na branchu master i zawartość pliku README.MD,
- Zmergeuj branch branch_B do brancha master i rozwiąż konflikty.

Usługi hostingowe



GitHub i Bitbucket

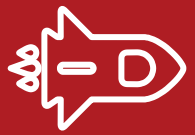
Usługi hostingowe
GitHub

- Najbardziej popularny hosting dla projektów Open Source,
- Darmowy dla projektów Open Source - repozytoria publiczne,
- Nielimitowane repozytoria prywatne - płatne,
- Zintegrowany issue tracker, wiki i narzędzie do zarządzania projektami i wiele innych...
- Przykładowe repozytorium - https://github.com/google/guava

Usługi hostingowe
BitBucket

- Darmowy dla małych zespołów (do 5 użytkowników),
- Dostępna opcja self-hosted,
- Doskonale integruje się z ekosystemem Atlassian - JIRA, Bamboo, Hipchat,
- Przykładowe repozytorium:
  - https://bitbucket.org/tmcmaster/library-generic-rest-crud/branches/

Pull request

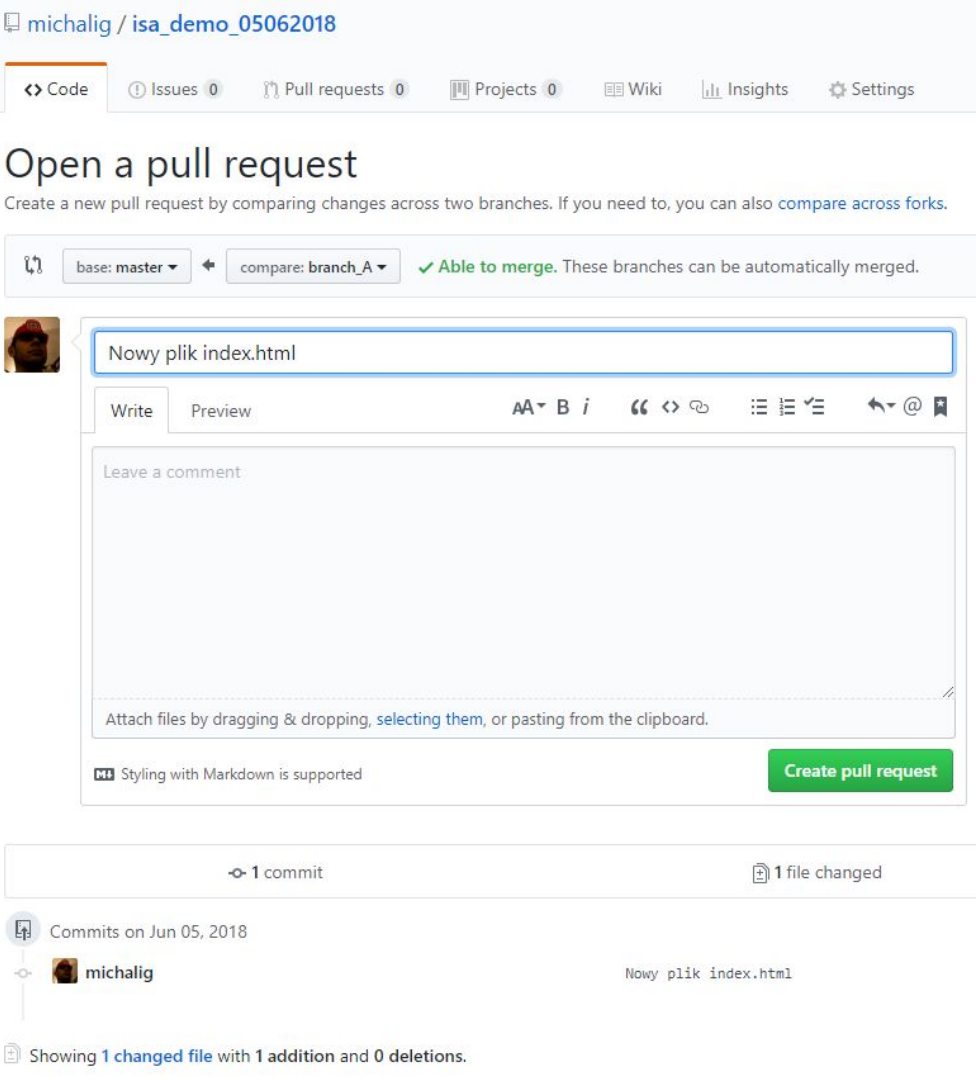


w GitHub

Pull request
Co to właściwie jest

- Pull request to prośba o zmergeowanie zmian z jednego brancha do drugiego,
- Serwisy takie jak GitHub pozwalają na tworzenie PRów, przegląd kodu, komentowanie i zatwierdzanie PR'ów,
  - Można skonfigurować GitHub'a tak, że kod nie może być mergeowany bez PR'a
- Każda zmiana trafiająca do repozytorium powinna przejść przez PR.

Pull request
demo



Pull request
zadanie

- Stwórz nowy branch w swoim repozytorium,
- Dokonaj w nim dowolnych zmian,
- Zacommituj i wypushuj zmiany,
- W GitHub stwórz PR dla tego brancha,
- Zobacz jakie są możliwości komentowania PRów,
- Zmergeuj PR z poziomu GitHub,
- Zrób pulla na lokalnym repozytorium i zobacz jak wygląda historia.

Dzieki!!

Pytania?

michal.ignaciuk@gmail.com