Branch i merge

1. Utwórz nowe repozytorium korzystając z komendy “git init” tak jak w poprzednim ćwiczeniu.

2. Utwórz plik tekstowy plik.txt z poniższą zawartością:

aaa
bbb
ccc

3. Wykonaj commit powyższego pliku (git add, git commit).

```
c:\dev\git-cwiczenia2>git add plik.txt

c:\dev\git-cwiczenia2>git commit -m "dodano plik.txt"
[master (root-commit) d366080] dodano plik.txt
 1 file changed, 3 insertions(+)
 create mode 100644 plik.txt
```

4. Utwórz nowy branch o nazwie “testowy”:

```
c:\dev\git-cwiczenia2>git branch testowy
```

5. Przejdź na nowo utworzony branch:

```
c:\dev\git-cwiczenia2>git checkout testowy
Switched to branch 'testowy'
```

6. Dopisz kolejną linię (“ddd”) do pliku tekstowego. Wykonaj commit.

```
c:\dev\git-cwiczenia2>git add plik.txt

c:\dev\git-cwiczenia2>git commit -m "zmieniono plik.txt"
[testowy 1417a70] zmieniono plik.txt
 1 file changed, 2 insertions(+), 1 deletion(-)
```

7. Przejdź z powrotem na branch master. Sprawdź zawartość pliku ostatnia zmiana nie jest widoczna, ponieważ była wykonana na innym branchu. Podobnie komenda “git log” nie pokaże commitów wykonanych na innych branchach niż aktualny (czyli

master).

```
c:\dev\git-cwiczenia2>git checkout master
Switched to branch 'master'
```

```
c:\dev\git-cwiczenia2>git log
commit 152d9e046263c2dba587d561583ae43d7e82a1ae
Author: cezary holub <cezary.holub@cohesiva.com>
Date:   Fri Mar 27 21:58:14 2015 +0100

    dodano plik.txt
```

8. Wykonaj merge brancha “testowy”:

```
c:\dev\git-cwiczenia2>git merge testowy
Updating 152d9e0..1417a70
Fast-forward
 plik.txt | 3 ++-
 1 file changed, 2 insertions(+), 1 deletion(-)
```

Używając komendy “git merge” podajemy nazwę brancha który chcemy połączyć z aktualnie aktywnym branchem (czyli tym, który wyświetli się po użyciu “git status”).

9. Branch “testowy” był o 1 commit dalej niż branch “master”. Na branchu “master” nie było żadnych innych zmian od punktu rozgałęzienia, więc efektem merga jest po prostu dodanie ostatniego commita do brancha “master”. Zweryfikuj to komendą “git log” wynik powinien być identyczny na obu branchach.

Branch master:

```
c:\dev\git-cwiczenia2>git log
commit 1417a7013f40f37b593b10b0c5f57db253d669b9
Author: cezary holub <cezary.holub@cohesiva.com>
Date:   Fri Mar 27 21:59:12 2015 +0100

    zmieniono plik.txt

commit 152d9e046263c2dba587d561583ae43d7e82a1ae
Author: cezary holub <cezary.holub@cohesiva.com>
Date:   Fri Mar 27 21:58:14 2015 +0100

    dodano plik.txt
```

Branch testowy:

```
c:\dev\git-cwiczenia2>git checkout testowy
Switched to branch 'testowy'

c:\dev\git-cwiczenia2>git log
commit 1417a7013f40f37b593b10b0c5f57db253d669b9
Author: cezary holub <cezary.holub@cohesiva.com>
Date:   Fri Mar 27 21:59:12 2015 +0100

    zmieniono plik.txt

commit 152d9e046263c2dba587d561583ae43d7e82a1ae
Author: cezary holub <cezary.holub@cohesiva.com>
Date:   Fri Mar 27 21:58:14 2015 +0100

    dodano plik.txt
```

10. Ponownie przełącz się na branch “testowy” i dodaj kolejną linię (“eee”). Wykonaj commit.

```
c:\dev\git-cwiczenia2>git checkout testowy
Already on 'testowy'

c:\dev\git-cwiczenia2>git add plik.txt

c:\dev\git-cwiczenia2>git commit -m "eee plik.txt"
[testowy 69c98ed] eee plik.txt
 1 file changed, 2 insertions(+), 1 deletion(-)
```

11. Wróć do brancha “master”. Utwórz nowy plik tekstowy z dowolną zawartością. Wykonaj commit.

```
c:\dev\git-cwiczenia2>git checkout master
Switched to branch 'master'

c:\dev\git-cwiczenia2>git add plik2.txt

c:\dev\git-cwiczenia2>git commit -m "nowy plik2.txt"
[master 9c4ed1c] nowy plik2.txt
 1 file changed, 1 insertion(+)
 create mode 100644 plik2.txt
```

12. W tym momencie od ostatniego mergea zarówno na branchu “master” jak i “testowy” wykonany został commit. Z tego powodu w wyniku “git merge” dodany zostanie dodatkowy commit łączący oba branche. Jego opis można podać używając flagi m jak poniżej:

```
c:\dev\git-cwiczenia2>git merge testowy -m "merge testowego do mastera"
Merge made by the 'recursive' strategy.
 plik.txt | 3 ++-
 1 file changed, 2 insertions(+), 1 deletion(-)
```

13. Zbadaj wynik operacji za pomocą “git log --graph”. Branch master posiada teraz zmiany z brancha testowy, ale połączenie branchy wymagało dodatkowego commita.

Dodanie flagi graph do komendy “git log” lepiej obrazuje sytuację:

```
c:\dev\git-cwiczenia2>git log --graph
*   commit 3a95fb36b4422b54ad2937e23af28f81afeefd7c
|\  Merge: 9c4ed1c 69c98ed
| | Author: cezary holub <cezary.holub@cohesiva.com>
| | Date:   Fri Mar 27 22:11:32 2015 +0100
| |
| |     merge testowego do mastera
| |
| * commit 69c98ed43e0598f8b79465c70ffe6856f73b9a8d
| | Author: cezary holub <cezary.holub@cohesiva.com>
| | Date:   Fri Mar 27 22:07:04 2015 +0100
| |
| |     eee plik.txt
| |
* | commit 9c4ed1c19e0aec95daa589943faa43669843be4f
|/  Author: cezary holub <cezary.holub@cohesiva.com>
|   Date:   Fri Mar 27 22:09:05 2015 +0100
|
|       nowy plik2.txt
|
* commit 1417a7013f40f37b593b10b0c5f57db253d669b9
| Author: cezary holub <cezary.holub@cohesiva.com>
| Date:   Fri Mar 27 21:59:12 2015 +0100
|
|     zmieniono plik.txt
|
* commit 152d9e046263c2dba587d561583ae43d7e82a1ae
  Author: cezary holub <cezary.holub@cohesiva.com>
  Date:   Fri Mar 27 21:58:14 2015 +0100

      dodano plik.txt
```

14. W poprzednim przypadku commit mergeujący mógł być wykonany automatycznie ponieważ zmiany na obu branchach dotyczyły różnych plików. Zmiany w tym samym pliku często również mogą być łączone automatycznie o ile nie są wykonane w tej samej linii. W takim przypadku mamy konflikt, który trzeba rozwiązać ręcznie. Aby zasymulować konfliktową sytuację, przełącz się na branch “testowy”. Dopisz do pierwszego pliku kolejną linię “fff” i wykonaj commit. Wróć na branch “master” i tym samym pliku dopisz linię “111”. Wykonaj commit.

```
c:\dev\git-cwiczenia2>git checkout testowy
Switched to branch 'testowy'

c:\dev\git-cwiczenia2>git add plik.txt

c:\dev\git-cwiczenia2>git commit -m "fff plik.txt"
[testowy 54af622] fff plik.txt
 1 file changed, 2 insertions(+), 1 deletion(-)

c:\dev\git-cwiczenia2>git checkout master
Switched to branch 'master'

c:\dev\git-cwiczenia2>git add plik.txt

c:\dev\git-cwiczenia2>git commit -m "111 plik.txt"
[master c5f92a1] 111 plik.txt
 1 file changed, 2 insertions(+), 1 deletion(-)
```

15. Spróbuj wykonać merge brancha “testowy”. Pojawi się informacja o konflikcie:

```
c:\dev\git-cwiczenia2>git merge testowy
Auto-merging plik.txt
CONFLICT (content): Merge conflict in plik.txt
Automatic merge failed; fix conflicts and then commit the result.
```

16. Git jest w trybie rozwiązywania konfliktu. Wykonaj “git status” by poznać szczegóły:

```
c:\dev\git-cwiczenia2>git status
On branch master
You have unmerged paths.
  (fix conflicts and run "git commit")

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

        both modified:   plik.txt

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

Odpowiedź programu informuje nas w którym plik jest konflikt. W tym przypadku nie jest to zaskoczeniem, ale w rzeczywistym projekcie konflikty wynikają najczęściej gdy łączone są zmiany wykonane przez 2 różne osoby nie zawsze jest łatwe do przewidzenia gdzie i kiedy wystąpi konflikt.

17. Git oczekuje od użytkownika rozwiązania konfliktu (poprzez ręczną edycję pliku) po czym wykonania commita (będzie to merge commit tak jak poprzednio, z tą różnicą że nie może on być teraz wykonany automatycznie). Aby rozwiązać konflikt otwórz (np. w Notatniku) plik w którym on wystąpił. Jego zawartość będzie podobna do poniższej:

```
aaa
bbb
ccc
ddd
eee
<<<<<<< HEAD
111
=======
fff
>>>>>>> testowy
```

Znaczniki tekstowe pokazują obydwie wersje odpowiednio “111” z mastera (oznaczone jako HEAD, bo branch master jest aktualnie aktywny) oraz “fff” z testowego. Zadaniem użytkownika jest sprowadzenie pliku do żądanej postaci Czasami

chcemy po prostu wybrać jedną z wersji usuwamy wtedy pozostałe linie. Czasami obydwie zmiany wnoszą coś istotnego wtedy musimy sami je połączyć w żądany tekst. Na potrzeby tego ćwiczenia załóżmy że obydwie zmiany są dla nas istotne, przy czym "fff" ma być przed "111". Plik powinien mieć zatem taką zawartość:

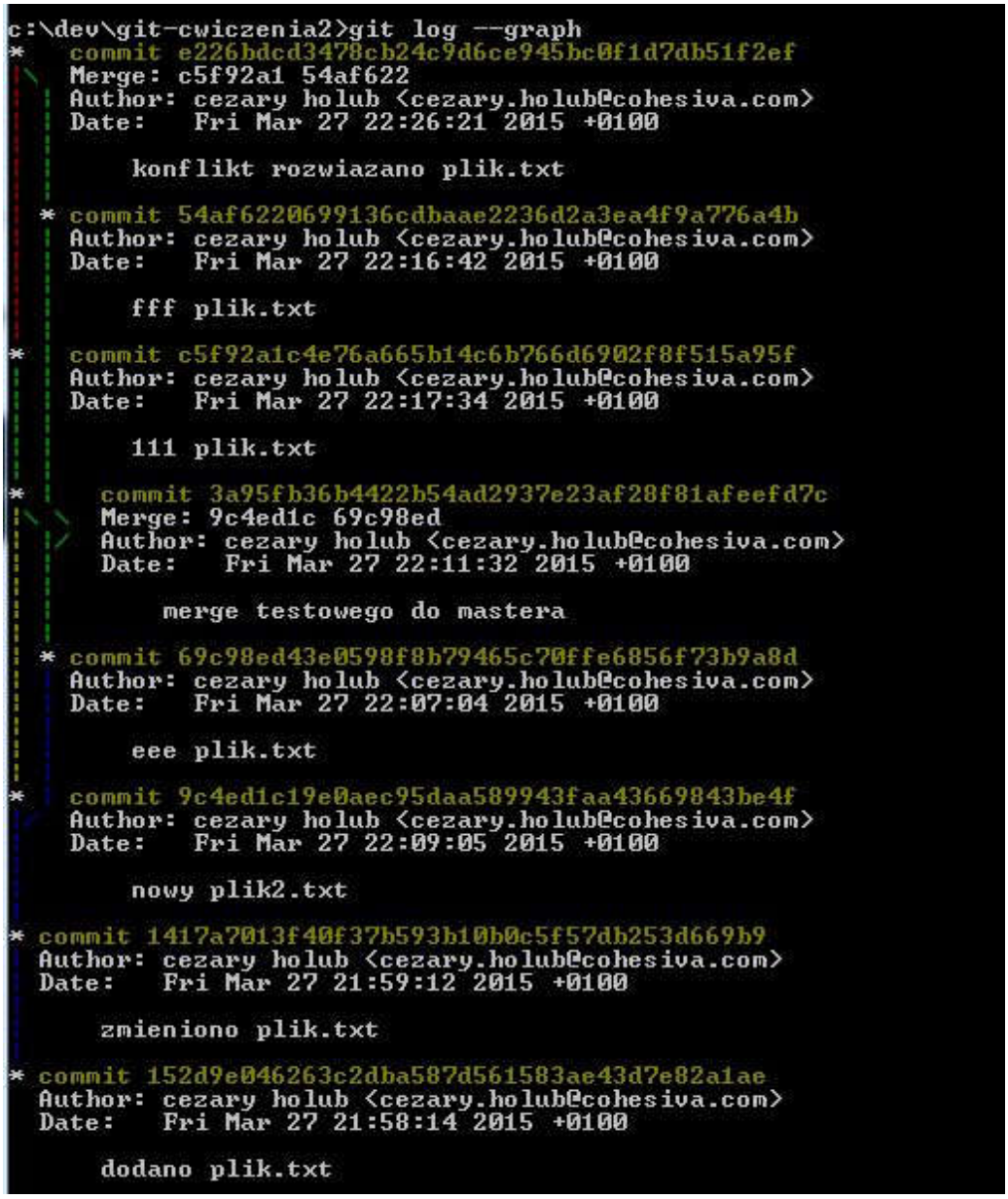
aaa
bbb
ccc
ddd
eee
fff
111

18. Po zapisaniu pliku wykonaj commit standardowymi komendami (git add . oraz git commit -m "opis").

```
c:\dev\git-cwiczenia2>git add plik.txt

c:\dev\git-cwiczenia2>git commit -m "konflikt rozwiazano plik.txt"
[master e226bdc] konflikt rozwiazano plik.txt
```

19. Zbadaj efekt komendą "git log graph"



Praca z repozytorium zdalnym (na ocenę)

Następne laboratorium