Systemy kontroli wersji

Git
Część II – Gałęzie

Aleksander Lamża
ZKSB · Instytut Informatyki
Uniwersytet Śląski w Katowicach

aleksander.lamza@us.edu.pl

Zawartość

– Gałęzie

Wstępne wymagania

– Wprowadzenie do systemów kontroli wersji

– Podstawy Git-a

Gałęzie

Dlaczego?

Gałęzie

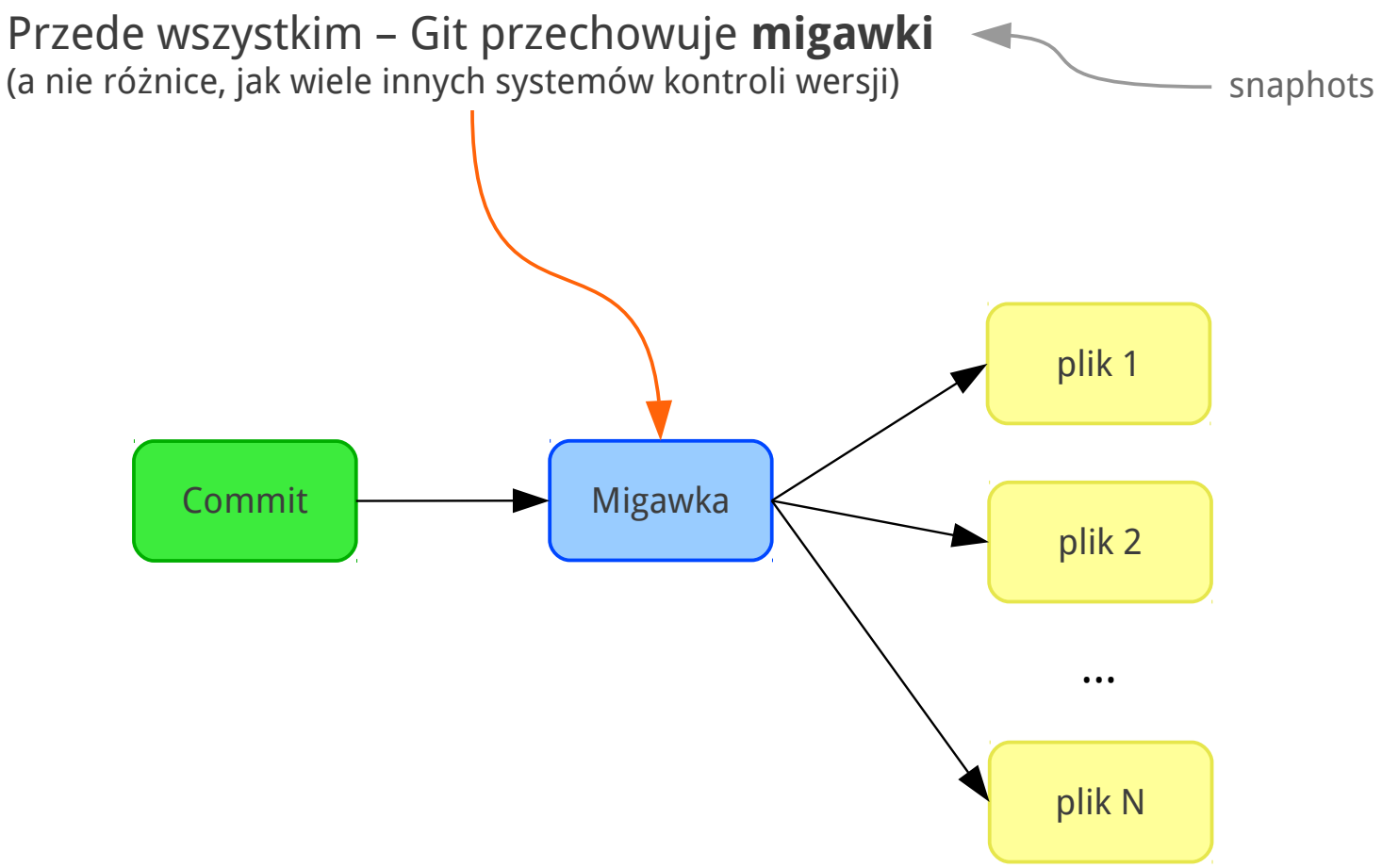
Dlatego, że dzięki nim można pracować nad oprogramowaniem

bez potrzeby **„mieszania” w głównej linii rozwoju**

i bez częstego **cofania wprowadzonych zmian**.

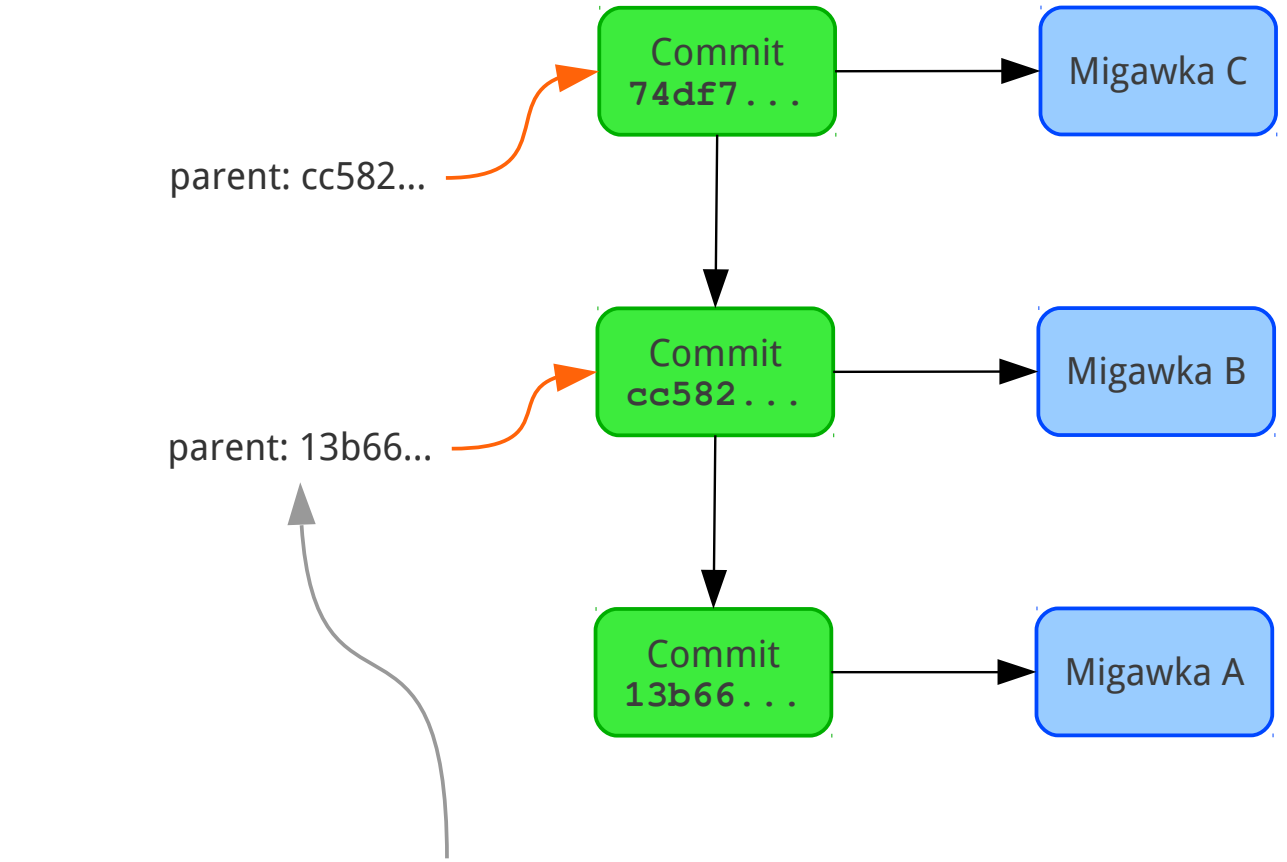
Gałęzie – zasada działania

Aby w pełni zrozumieć, jak działają gałęzie, musimy poznać wewnętrzną strukturę repozytorium Git-a.



Gałęzie – zasada działania

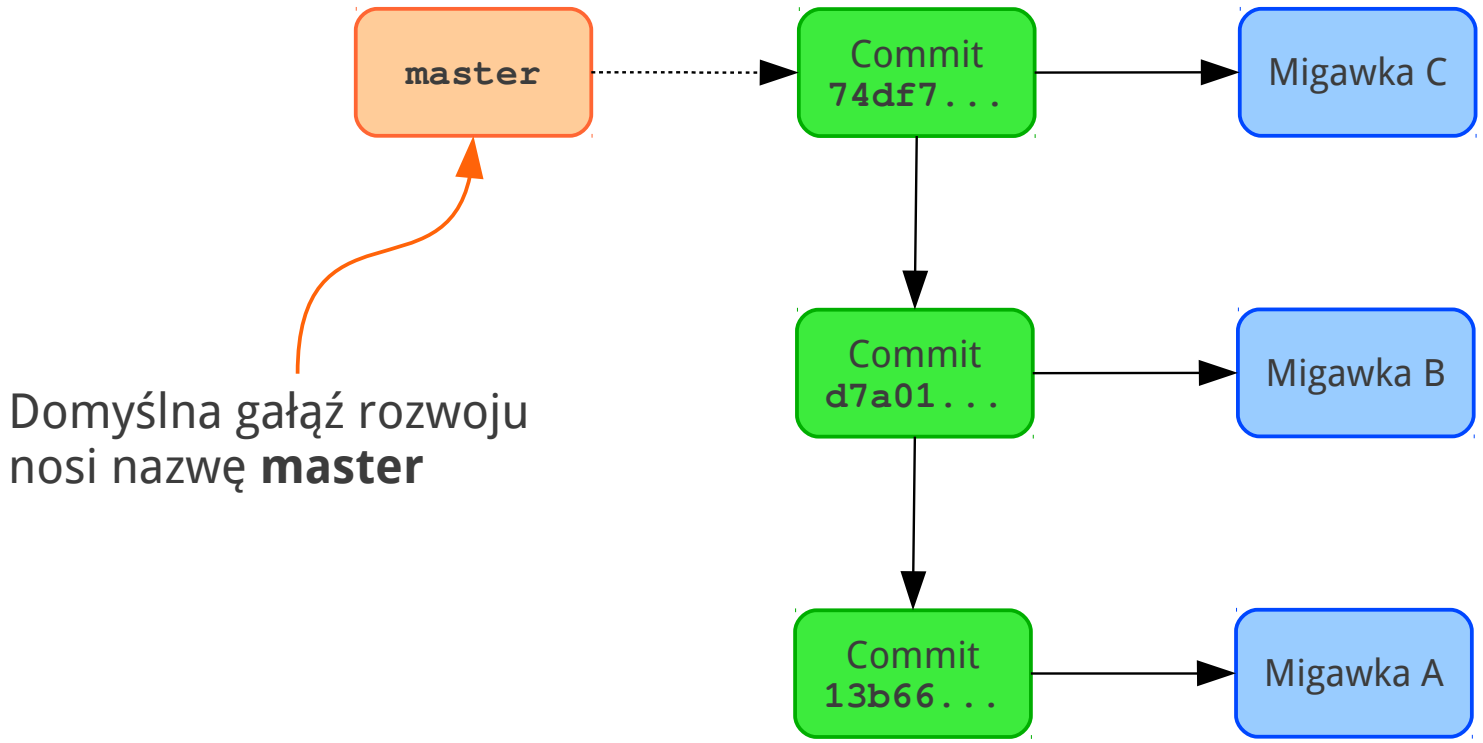
Przy wielu commitach sytuacja wygląda tak:



commity (i nie tylko) są oznaczane haszami SHA-1
(np. cc582b94cc2d1595e00508223858a7daa93ac3f6)

Gałęzie – zasada działania

Gałąź jest **wskaźnikiem na commit**

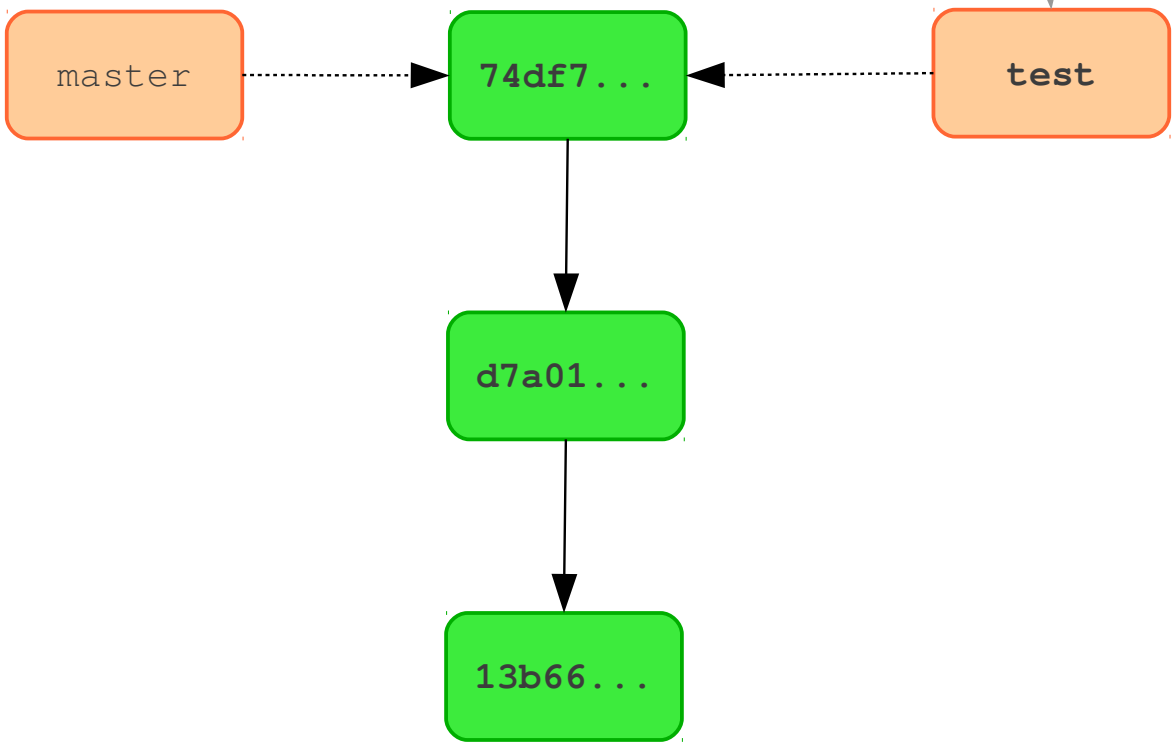


Gałęzie – zasada działania

Co się stanie, kiedy **utworzymy nową gałąź**?

Tworzenie gałęzi jest mało kosztowne (sprowadza się do utworzenia 41-bajtowego pliku)

```
git branch test
```

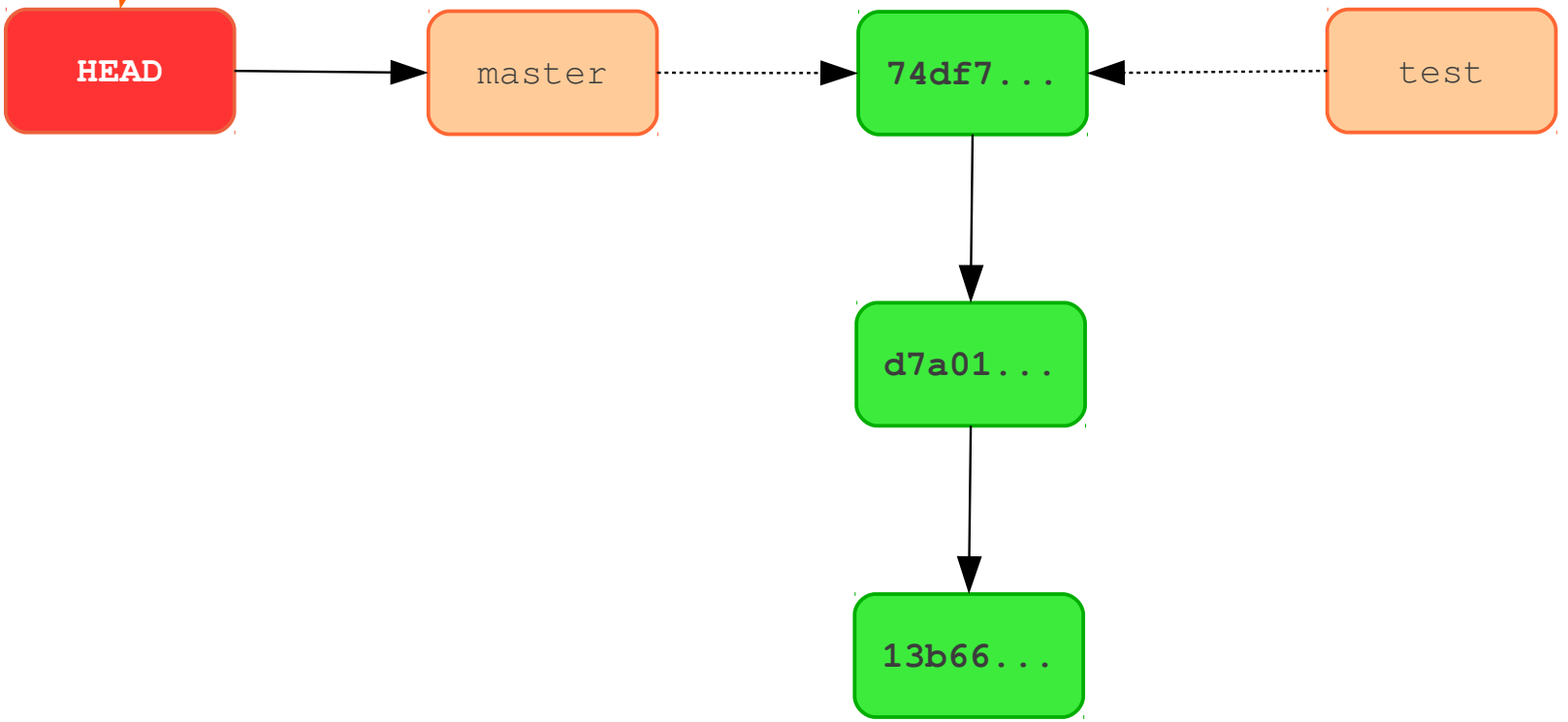


Gałęzie – zasada działania

Skąd wiedzieć, na której gałęzi się siedzi?

Służy do tego specjalny wskaźnik **HEAD**

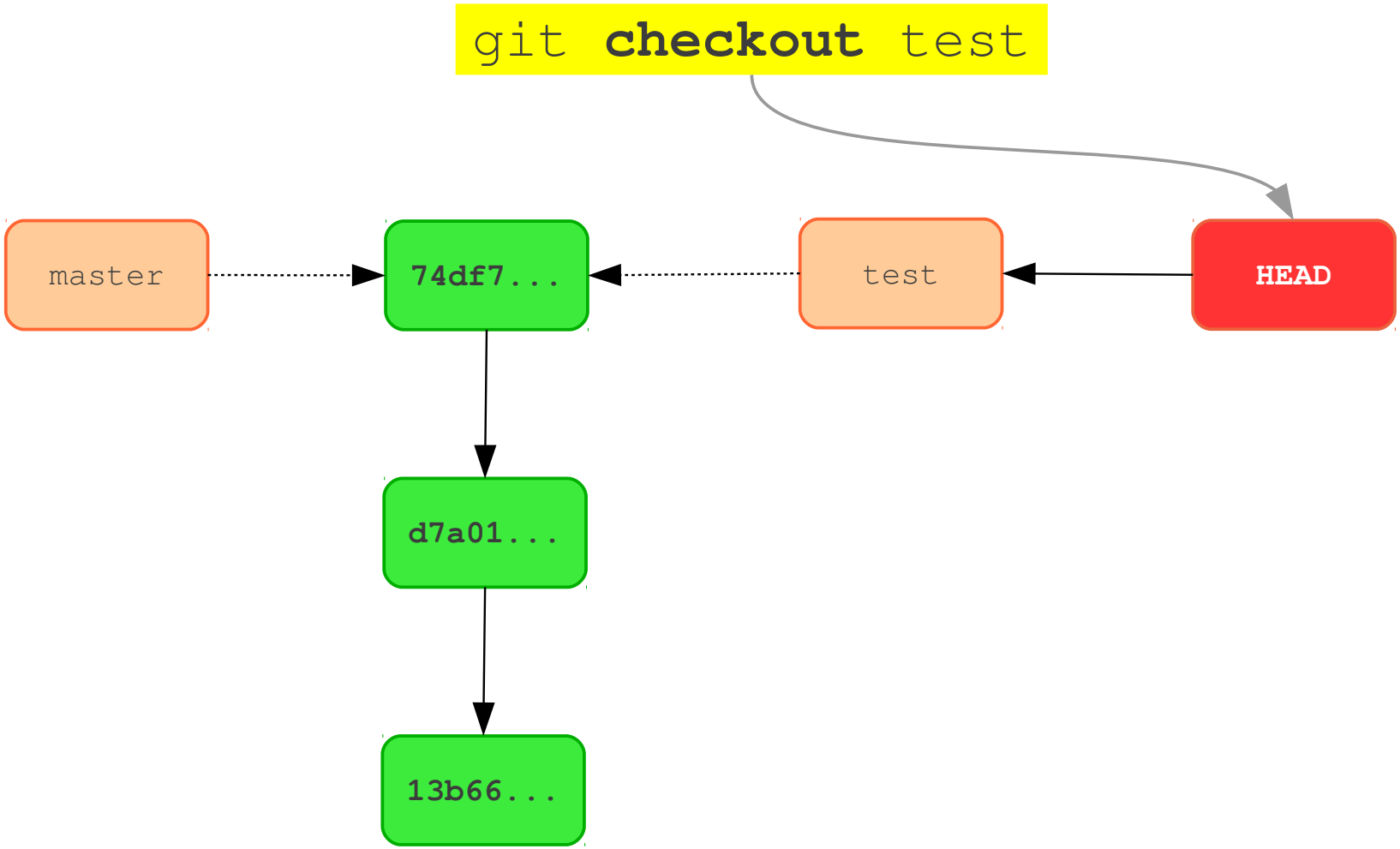
Uwaga! W innych systemach kontroli wersji (np. SVN) HEAD jest czymś trochę innym...



Gałęzie – zasada działania

Polecenie `git` **`branch`** jedynie tworzy gałąź, ale na nią nie przeskakuje.

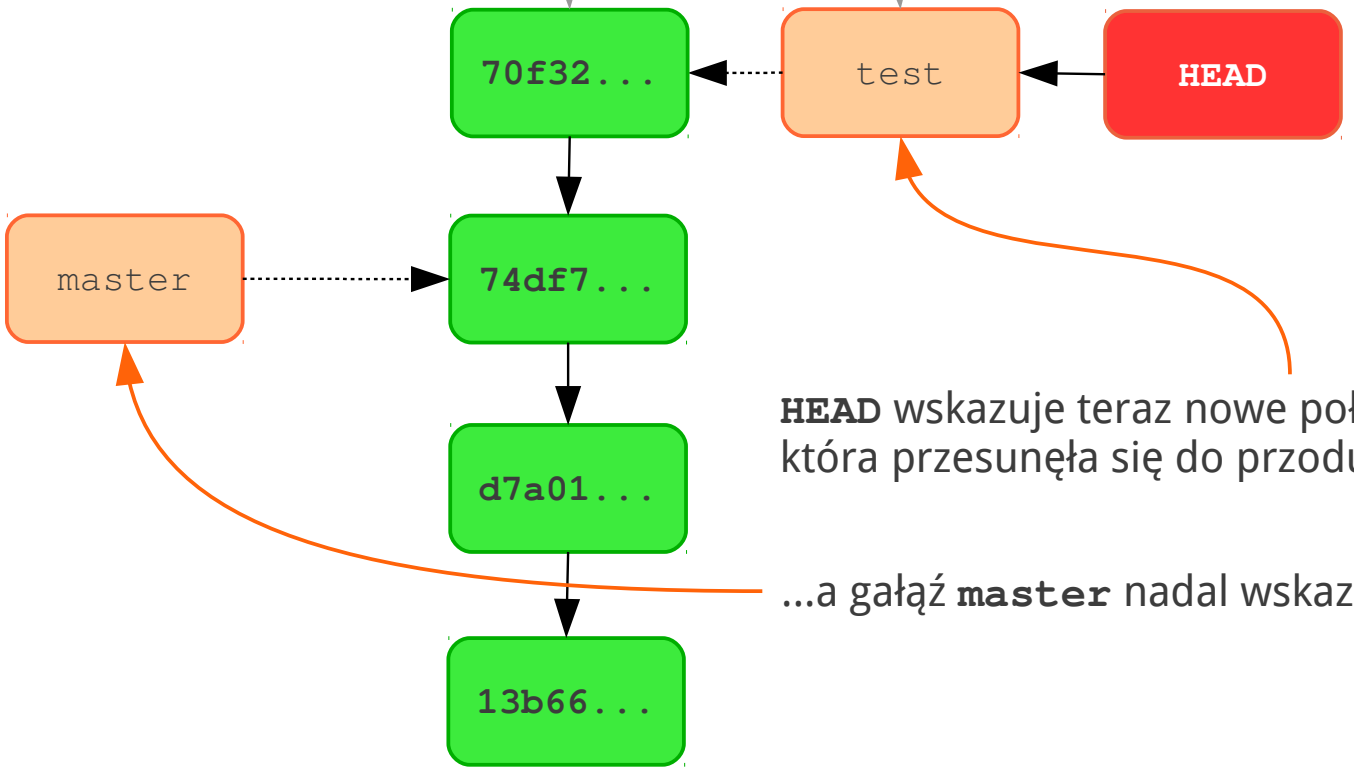
Do przeskakiwania na inne gałęzie służy polecenie



Gałęzie – zasada działania

Co się stanie, jeżeli w nowej gałęzi zrobimy commit?

```
$ git commit -m "zmiana w gałęzi test"
[test 70f32aa] zmiana w gałęzi test
...
```



HEAD wskazuje teraz nowe położenie gałęzi **test**, która przesunęła się do przodu...

...a gałąź **master** nadal wskazuje poprzedni commit.

Gałęzie – zasada działania

No dobra, a gdybym przeskoczył do gałęzi `master`...

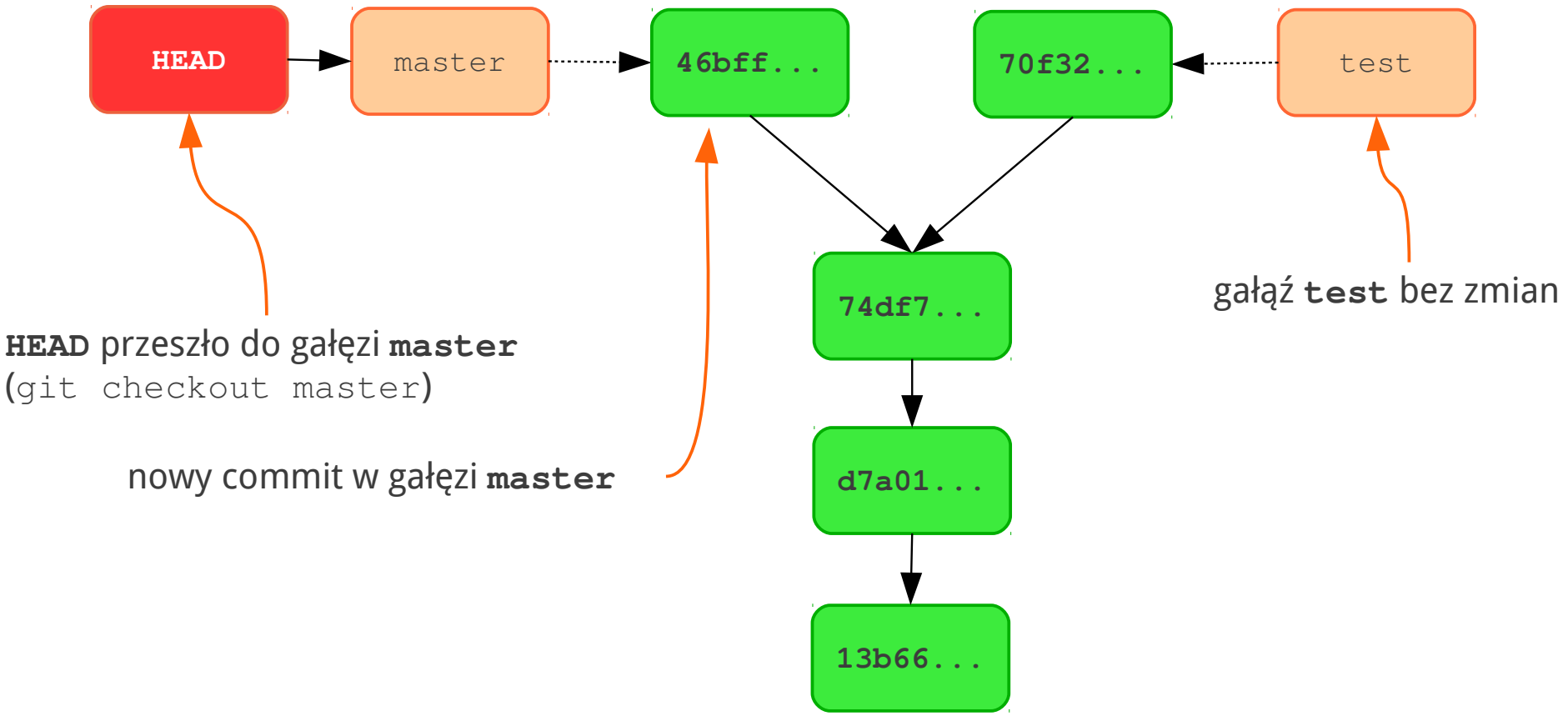
```
$ git checkout master
Switched to branch 'master'
```

...zmodyfikował plik i zrobił kolejny commit?

```
$ git commit -m "zmiana w gałęzi master"
[master 46bff37] zmiana w gałęzi master
...
```

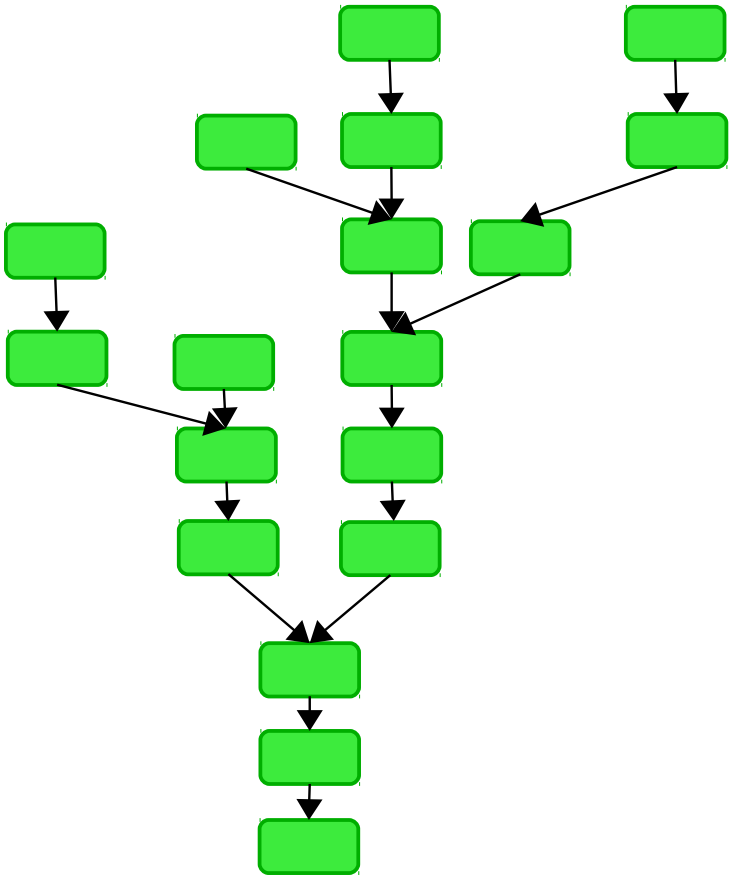
?

Gałęzie – zasada działania



Gałęzie – zasada działania

Jak widać, dzięki gałęziom można tworzyć całkiem rozbudowane struktury rozwoju...



Gałęzie

Gałęzie

Dzięki temu, że tworzenie gałęzi jest tak proste
(i mało kosztowne, więc szybkie),

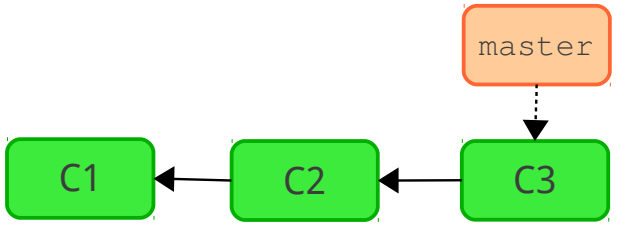
**podczas pracy nad projektem
można z nich intensywnie korzystać.**

a nawet **trzeba!**

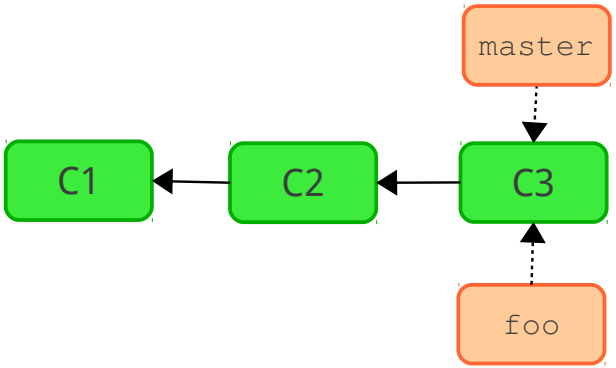
Przydałoby się kilka praktycznych przykładów, prawda?

Gałęzie w praktyce – proste scalanie

Jest sobie pewien projekt:



Twoim zadaniem na dziś jest napisanie klasy `Foo`.
Od czego zaczynasz? **Tworzysz nową gałąź.**




```
$ git branch foo
$ git checkout foo
Switched to branch 'foo'
```

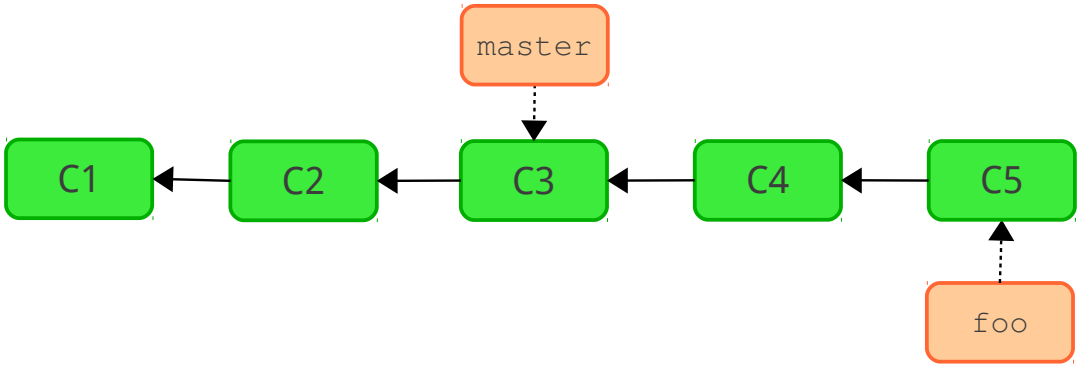
To samo można załatwić szybciej:
`git checkout -b foo`

Gałęzie w praktyce – proste scalanie

Tworzysz klasę i sobie nad nią pracujesz, od czasu do czasu robiąc commity:

Praca, praca
i jeszcze raz praca...

```
...
$ git commit -m "C4"
...
$ git commit -m "C5"
```



Gałęzie w praktyce – proste scalanie

Klasa `Foo` napisana i przetestowana! Co teraz? **Scalasz swoją gałąź z główną gałęzią.**

```
git merge foo
```

Najpierw przeskakujemy na `master`

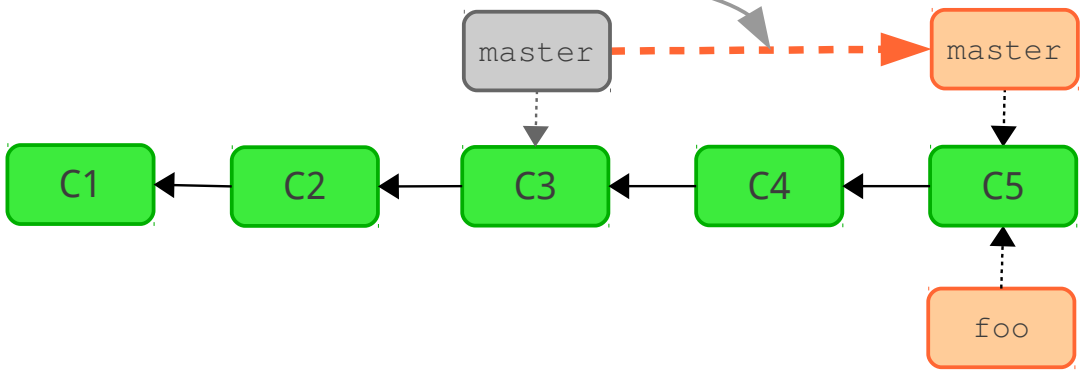
Później dołączamy zmiany z `foo`

```
$ git checkout master

$ git merge foo

Updating 2215776..fe74e58
Fast-forward
 Foo |    1 +
 1 file changed, 1
insertion(+)
 create mode 100644 Foo
```

„Fast forward” oznacza, że w celu scalenia wystarczyło przesunąć wskaźnik gałęzi **`master`** do przodu.



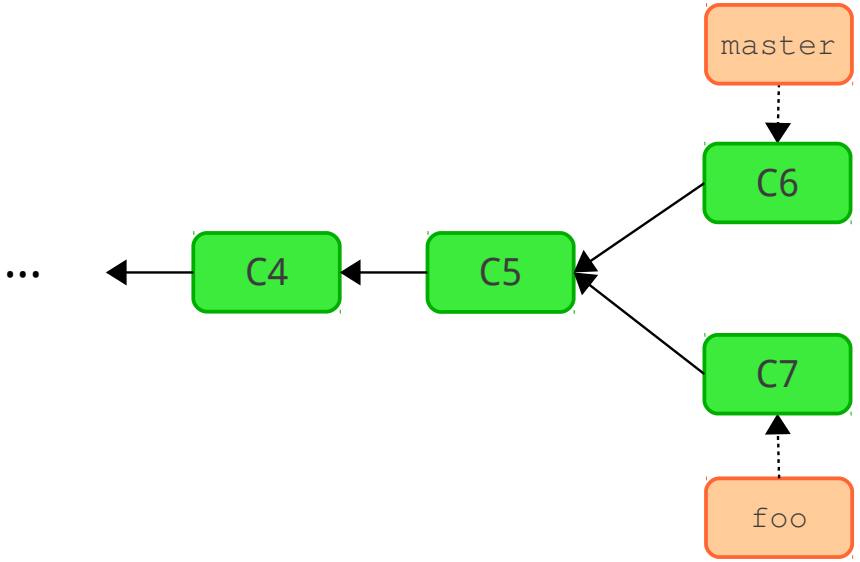
Gałęzie – scalanie

To scalanie gałęzi to bułka z masłem.

Ale chwila... Co by się stało,
gdyby podczas pracy w gałęzi foo
zmieniła się gałąź master?

Gałęzie w praktyce – scalanie trochę bardziej skomplikowane

Ech, te małpy – tylko by szukały problemów... No dobra, sytuacja wygląda tak:



Zmieniłeś coś w gałęzi `master`, a następnie zrobiłeś commit (C6).

Musiałeś jednak jeszcze coś zmienić w klasie `Foo`, więc przeskoczyłeś do gałęzi `foo`, wprowadziłeś zmiany i zrobiłeś commit (C7).

Teraz chcesz scalić:

Niby w porządku, ale wygląda inaczej...
Gdzie „fast forward”?

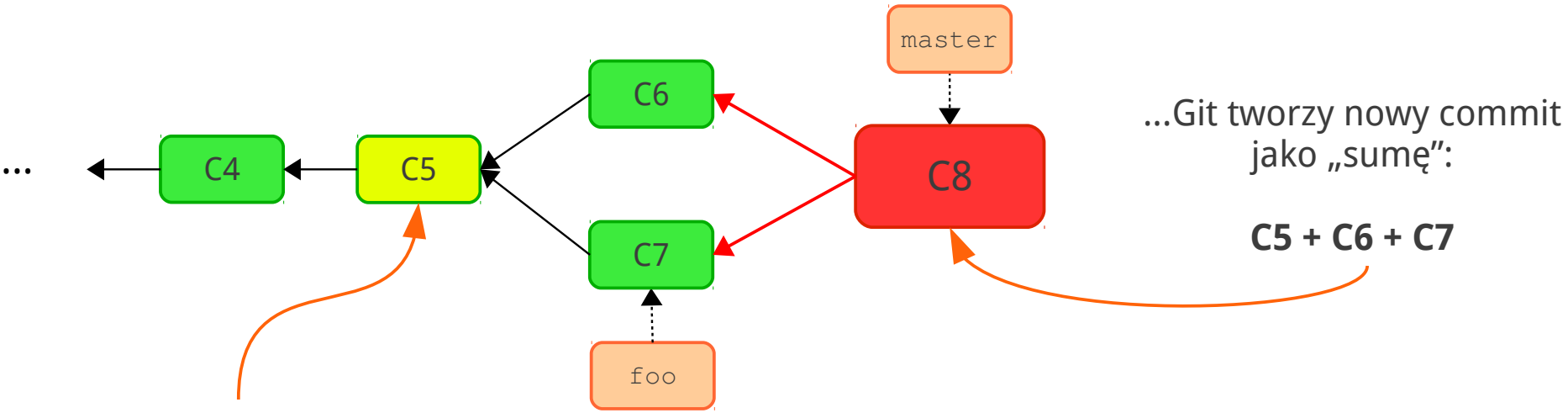
```
$ git checkout master

$ git merge foo

Merge made by the 'recursive' strategy.
 Foo |    2 +-
 1 file changed, 1 insertion(+), 1 deletion(-)
```

Gałęzie w praktyce – scalanie trochę bardziej skomplikowane

Nie da się „szybko przewinąć” wskaźnika `master`, bo commit C7 (z gałęzi `foo`) nie jest bezpośrednim potomkiem ostatniego commita z `master` (C6).



Uff...

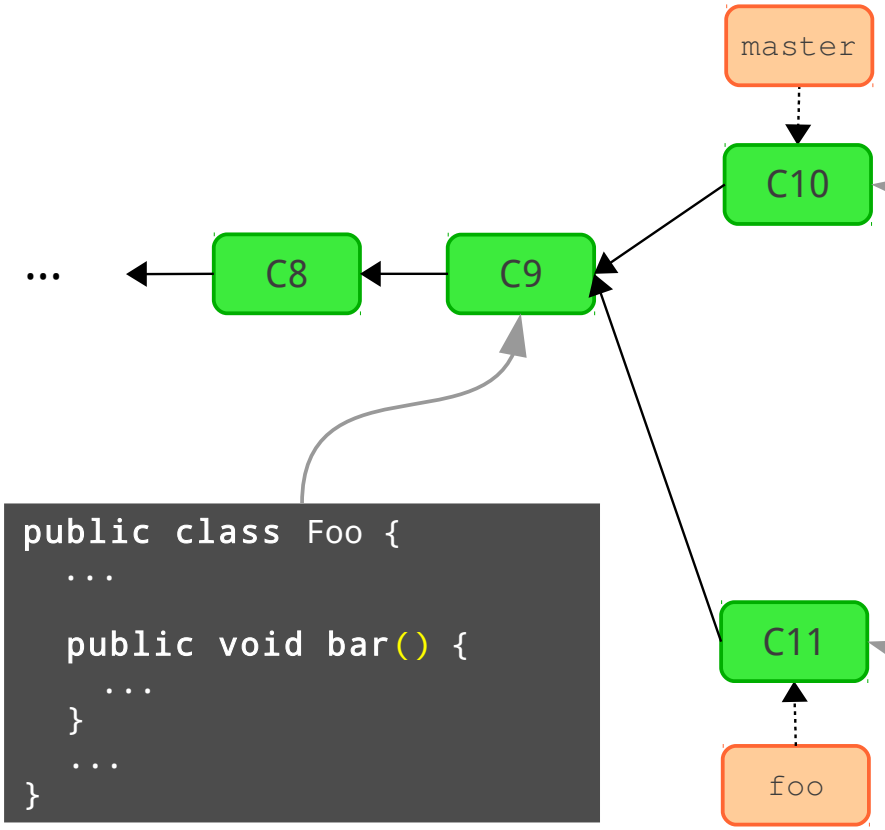
Dobra wiadomość jest taka, że wszystko poszło gładko.

Niestety jest też zła wiadomość – **nie zawsze wszystko idzie gładko**...

Gałęzie – scalanie

Czasem dochodzi do konfliktów...

Gałęzie – konflikty i ich rozwiązywanie



W gałęzi `master` w klasie `Foo` zmodyfikowałeś metodę `bar()` a następnie zrobiłeś commit (C10).

```
public class Foo {
  ...

  public void bar(int p) {
    ...
  }
  ...
}
```

```
public class Foo {
  ...

  public void bar() {
    ...
  }
  ...
}
```

W gałęzi `foo` również zmodyfikowałeś (ty lub ktoś inny) metodę `bar()` klasy `Foo`, a następnie zrobiłeś commit (C11).

```
public class Foo {
  ...

  public void bar(String p) {
    ...
  }
  ...
}
```

Teraz chcesz scalić zmiany z obu gałęzi

Gałęzie – konflikty i ich rozwiązywanie

No to scalamy:

```
$ git checkout master

$ git merge foo

Auto-merging Foo
CONFLICT (content): Merge conflict in Foo
Automatic merge failed; fix conflicts and then commit the result.
```

Git wykrył konflikt – nie udało się automatycznie połączyć zmian.

Plik Foo wygląda teraz tak:

Wszystko między `<<<<<< HEAD` i `=======` pochodzi z gałęzi `master` (tam był ustawiony HEAD)...

...a wszystko od `=======` do `>>>>>> foo` to zmiany z gałęzi `foo`.

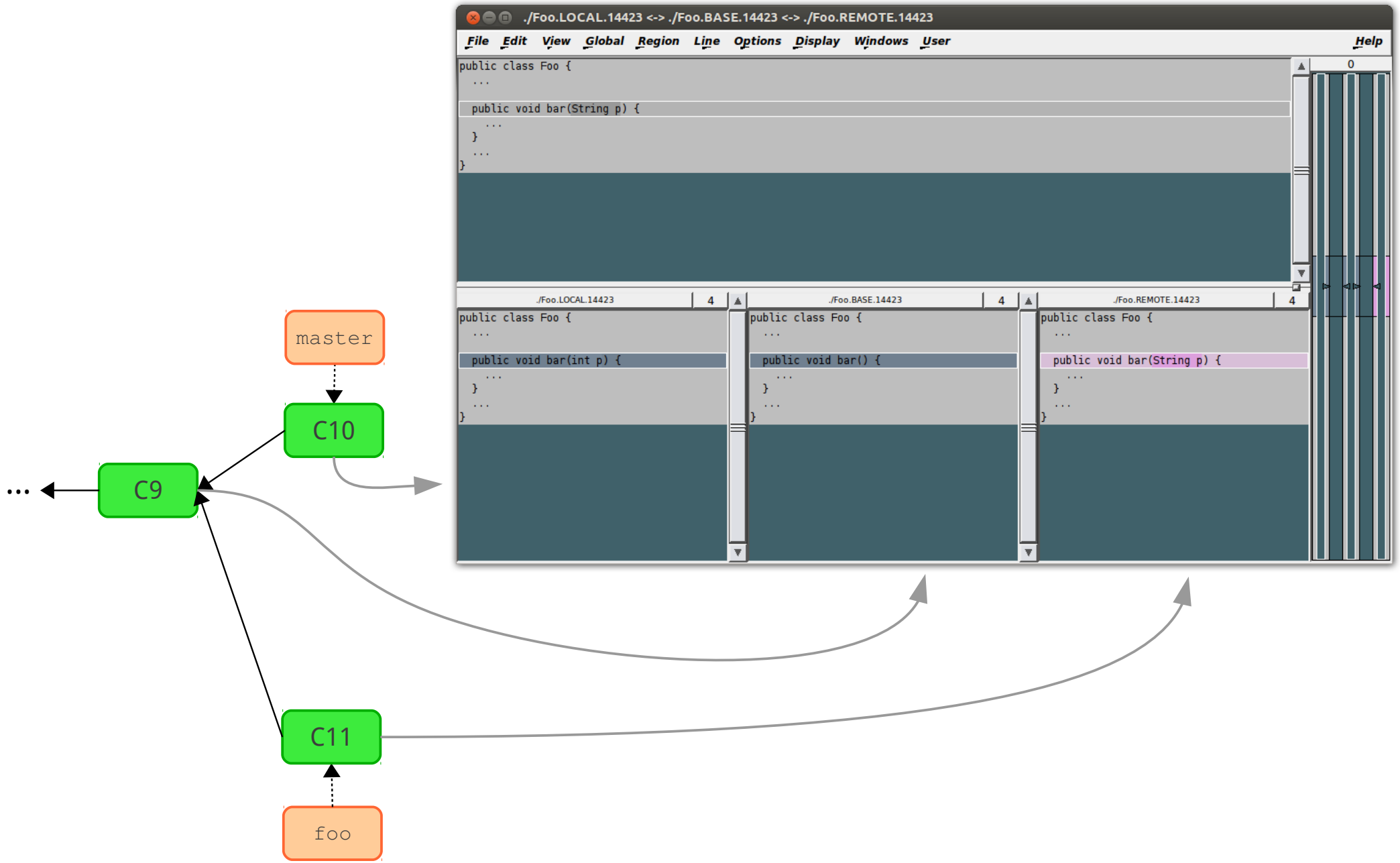
```
public class Foo {
  ...

<<<<<<< HEAD
  public void bar(int p) {
=======
  public void bar(String p) {
>>>>>>> foo
    ...
  }
  ...
}
```

Sami musimy się zająć scaleniem (poprzez edycję pliku), bo tego typu konfliktu Git nie rozwiąże.

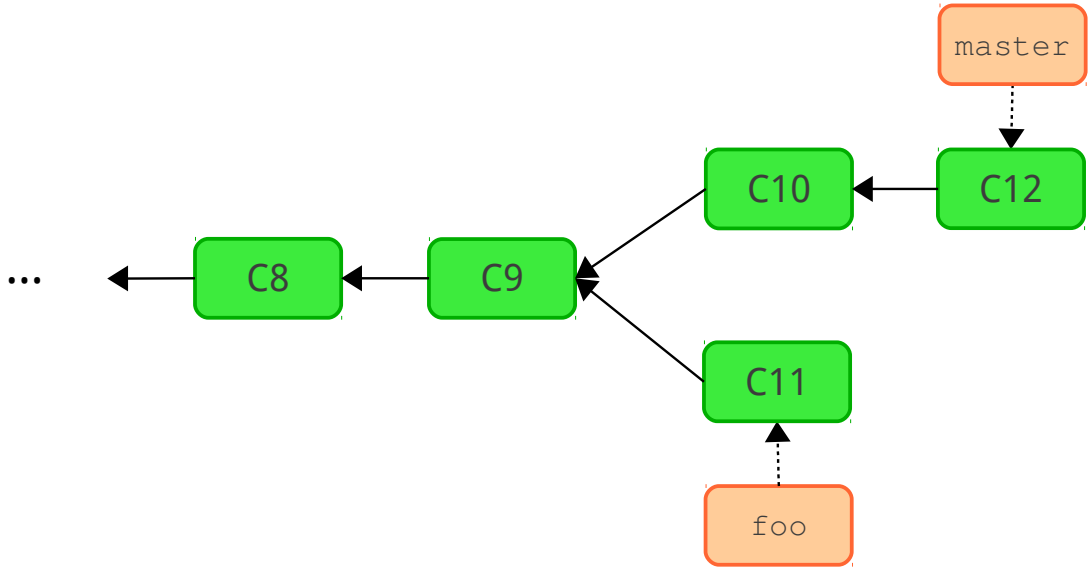
Gałęzie – konflikty i ich rozwiązywanie

Jest dostępnych kilka narzędzi, które mogą pomóc w ręcznym scalaniu, np. `xxdiff`:



Gałęzie – konflikty i ich rozwiązywanie

Po rozwiązaniu konfliktu trzeba ręcznie zrobić commit:



Gałęzie – zarządzanie gałęziami

Podczas pracy z gałęziami przydaje się kilka poleceń:

`git branch`

```
$ git branch

  foo
* master
```

Wyświetla listę gałęzi
(***** oznacza bieżącą)

`git branch -v`

```
$ git branch

  foo    9cc8e56 C11
* master 1dcd4d8 C12
```

Wyświetla listę gałęzi
wraz z commitami

`git branch --merged`

Wyświetla listę scalonych gałęzi
(dostępny jest też przełącznik
`--no-merged`)

Gałęzie – zarządzanie gałęziami

Gałęzie można też usuwać:

```
git branch -d foo
```

Gdyby w gałęzi zostały wprowadzone zmiany, ale nie zostały scalone, usuwanie by się nie powiodło:

```
$ git branch -d foo

error: The branch 'foo' is not an ancestor of your current HEAD.
```

Co dalej?

- Praca zespołowa
- GitHub