Git

Seweryn Kowalski

Październik 2021

Gałęzie Gita

- Git jako system kontroli wersji posiada wsparcie dla gałęzi
- Rozgałęzienie oznacza odbicie od głównego pnia linii rozwoju i kontynuację pracy
- Git zachęca do częstego rozgałęziania i scalania projektu, nawet kilkukrotnie w ciągu jednego dnia

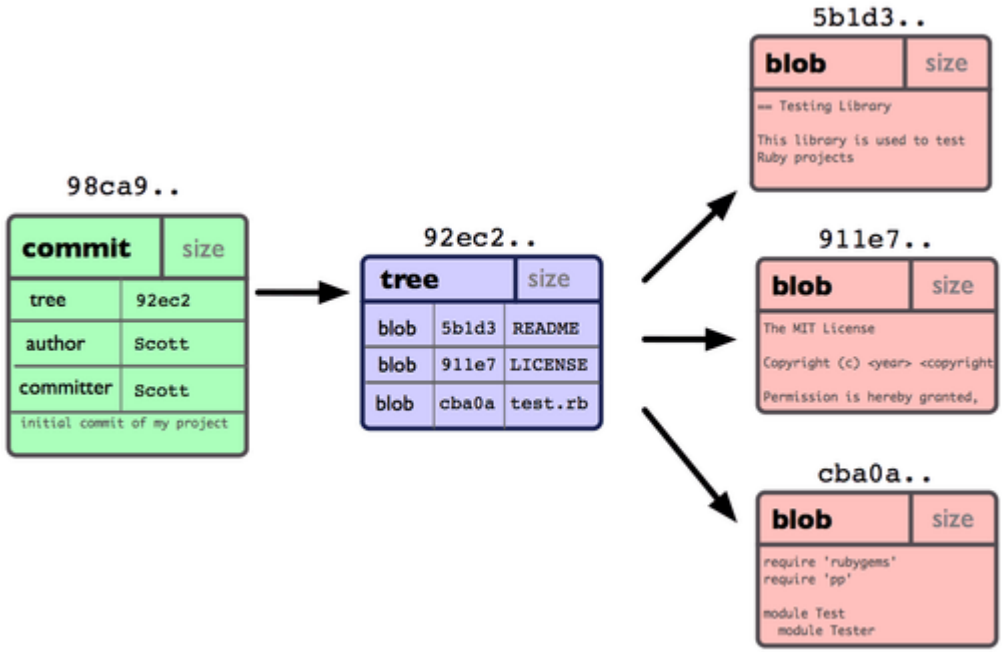
Czym jest gałąź

- Git nie przechowuje danych jako serii zmian i różnic, ale jako zestaw migawek.
- Kiedy zatwierdzasz zmiany w Gicie, ten zapisuje obiekt zmian (commit), który z kolei zawiera wskaźnik na migawkę zawartości, która w danej chwili znajduje się w poczekalni (staging area), metadane autora i opisu oraz zero lub więcej wskaźników na zmiany, które były bezpośrednimi rodzicami zmiany właśnie zatwierdzanej: brak rodziców w przypadku pierwszej, jeden w przypadku zwykłej, oraz kilka w przypadku zmiany powstałej wskutek scalenia dwóch lub więcej gałęzi.
- Gałęzie w Gicie są tak naprawdę prostymi plikami, zawierającymi 40 znaków sumy kontrolnej SHA-1 zestawu zmian, na który wskazują, są one bardzo tanie w tworzeniu i usuwaniu. Stworzenie nowej gałęzi zajmuje dokładnie tyle czasu, co zapisanie 41 bajtów w pliku (40 znaków + znak nowej linii).

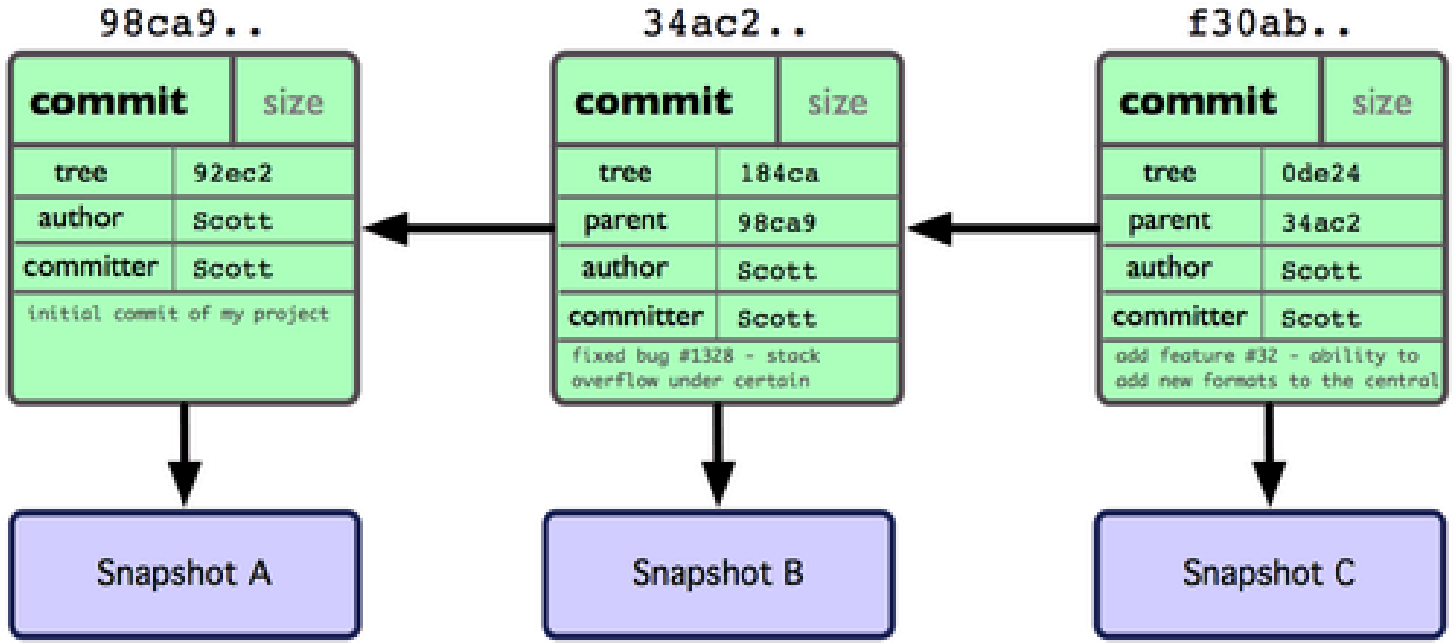
Dane repozytorium z jedną zatwierdzoną zmianą

$ git add README test.rb LICENSE

$ git commit -m 'Początkowa wersja mojego projektu'

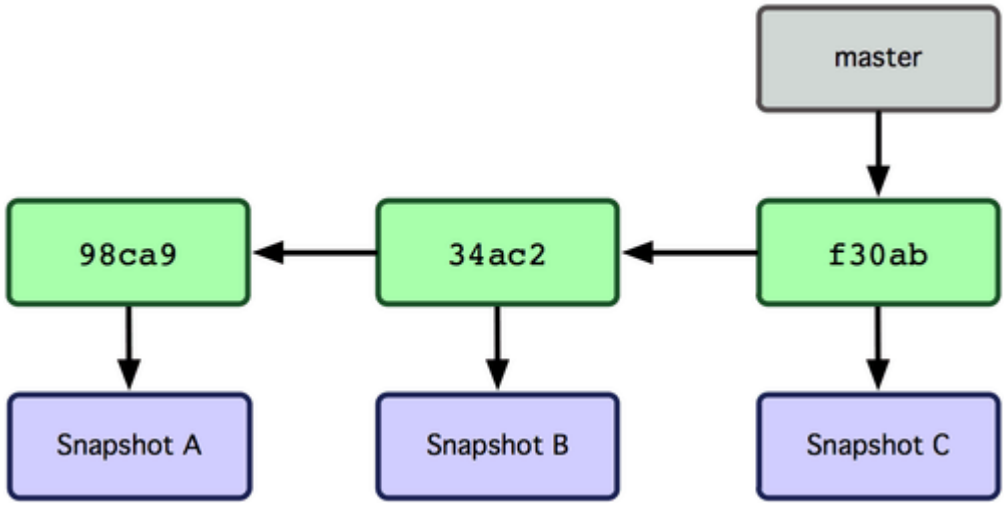


Dane Gita dla wielu zestawów zmian.



Gałąź w Gicie

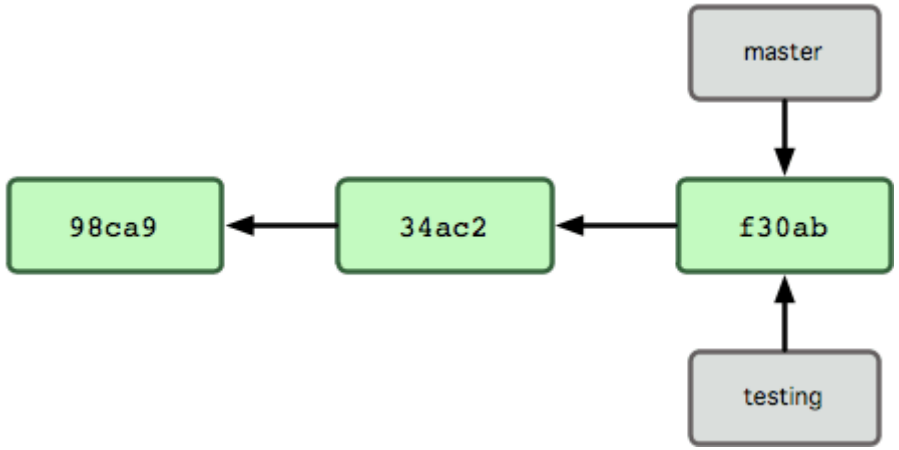
- Jest lekkim, przesuwalnym wskaźnikiem na któryś z owych zestawów zmian
- Domyślna nazwa gałęzi Gita to **master**
- Kiedy zatwierdzasz pierwsze zmiany, otrzymujesz gałąź master, która wskazuje na ostatni zatwierdzony przez Ciebie zestaw
- Z każdym zatwierdzeniem automatycznie przesuwa się ona do przodu.



Tworzenie nowej gałęzi

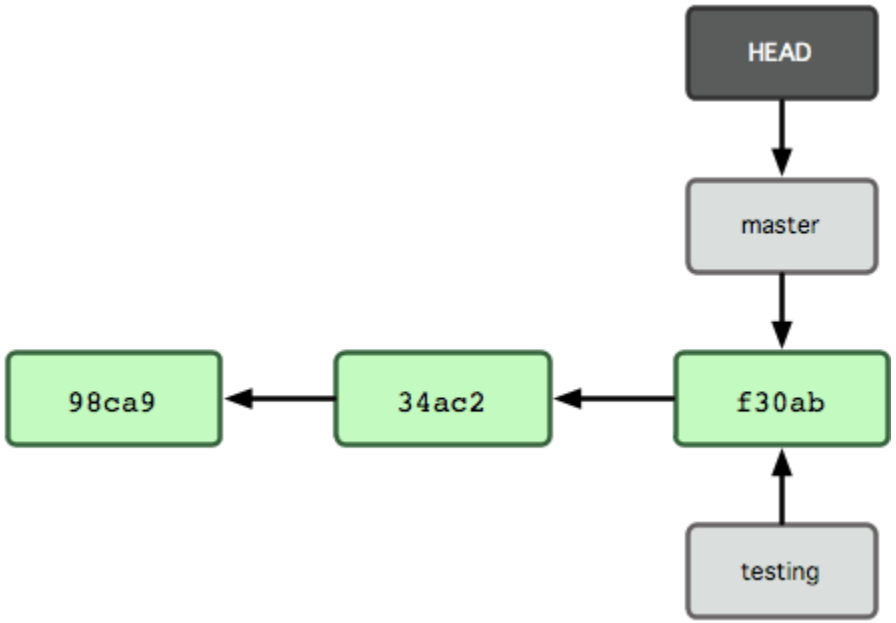
$ git branch testing

- Polecenie to tworzy nowy wskaźnik na ten sam zestaw zmian, w którym aktualnie się znajdujesz



- To gdzie się znajdujesz utrzymuje specjalny wskaźnik o nazwie HEAD
- Ważne – git **branch** tworzy nowa gałąź ale nie przełącza na nią

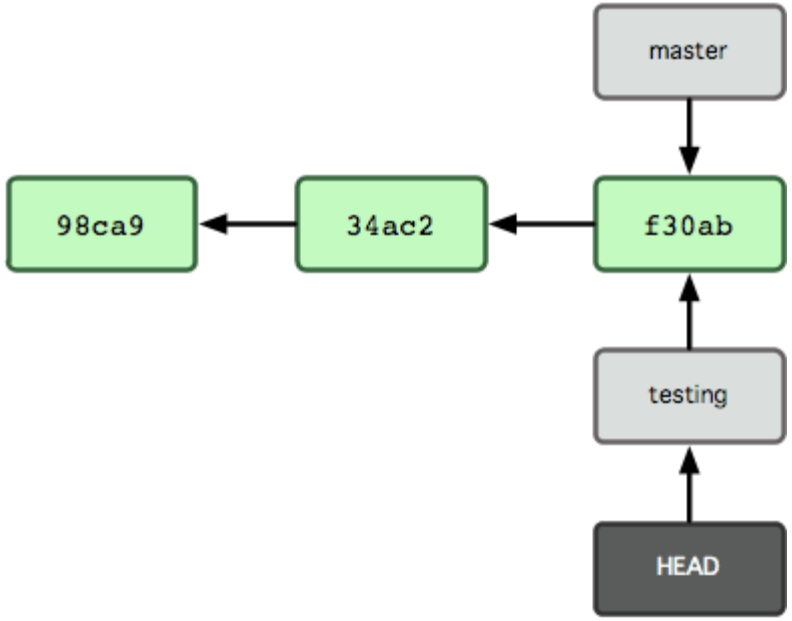
Tworzenie nowej gałęzi



Przełączanie na inną gałąź

$ git checkout testing

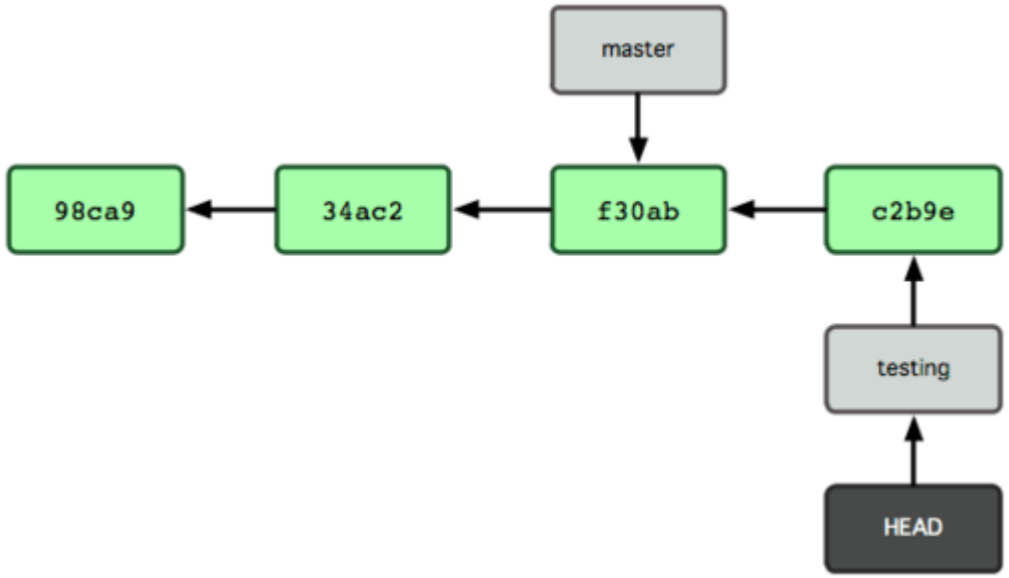
- HEAD zostaje zmieniony tak, by wskazywać na gałąź testing



Zmiany na gałęzi

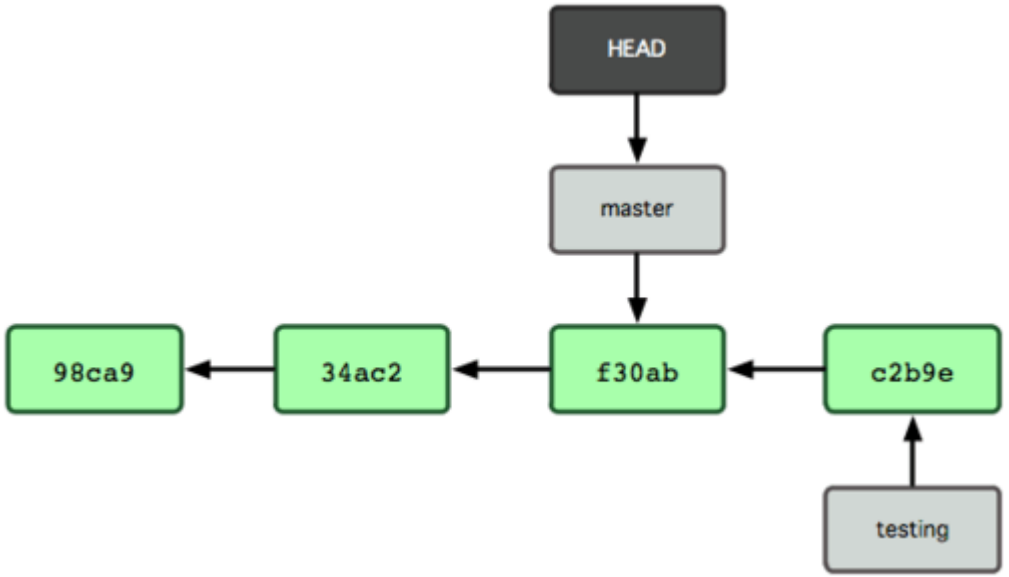
$ vim test.rb

$ git commit -a -m 'zmiana'



Wykonanie **checkout**, HEAD przesuwa się na inną gałąź

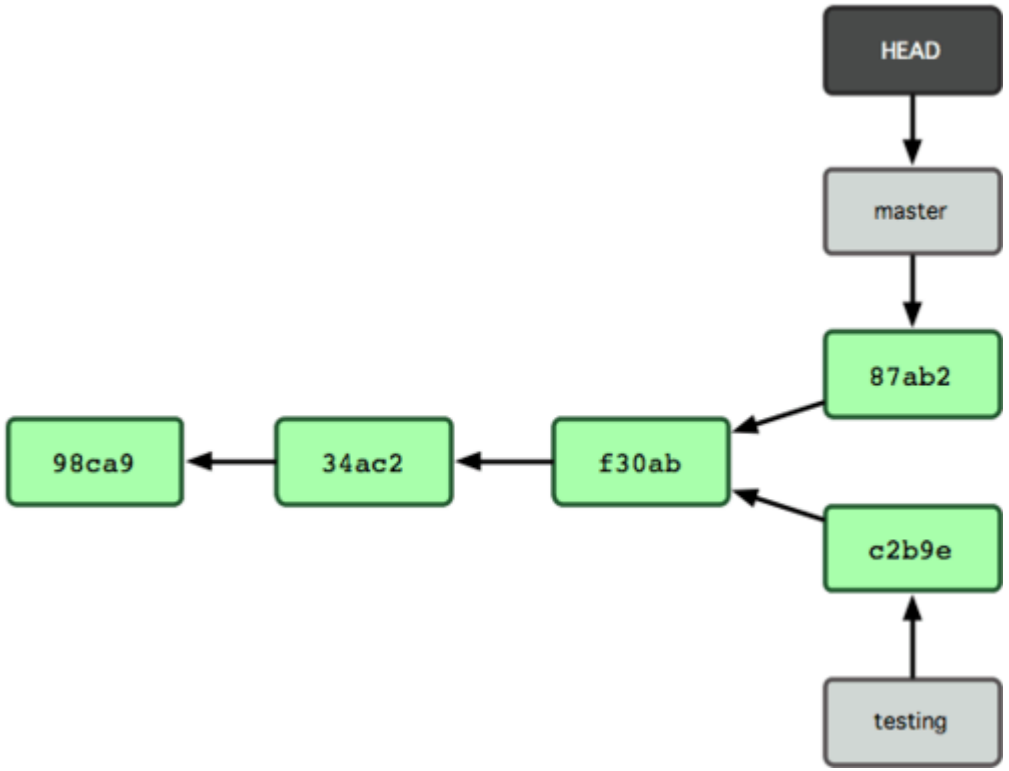
$ git checkout master



Rozwidlona historia gałęzi

$ vim test.rb

$ git commit -a -m 'inna zmiana'



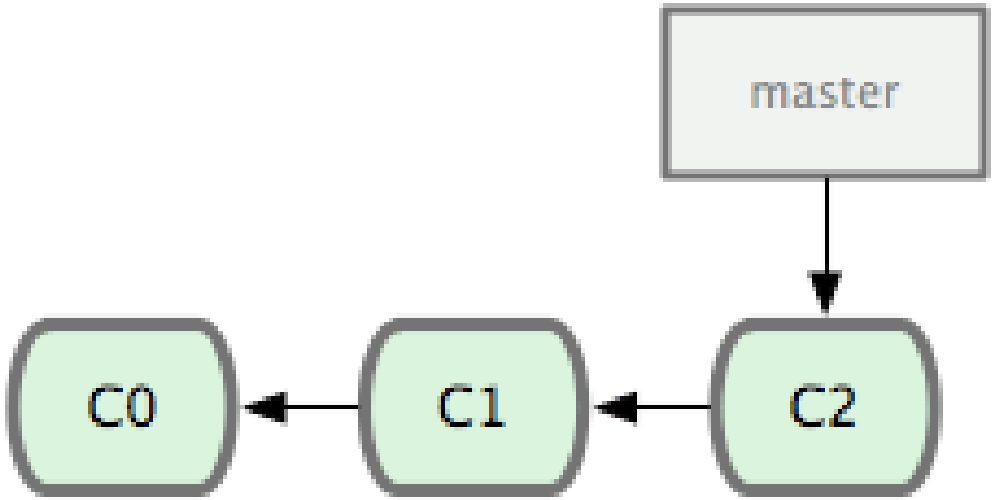
Scenariusz

- Wykonasz pracę nad stroną internetową.
- Stworzysz gałąź dla nowej funkcji, nad którą pracujesz.
- Wykonasz jakąś pracę w tej gałęzi.

Teraz otrzymasz telefon, że inny problem jest obecnie priorytetem i potrzeba błyskawicznej poprawki. Oto, co robisz:

- Powrócisz na gałąź produkcyjną.
- Stworzysz nową gałąź, by dodać tam poprawkę.
- Po przetestowaniu, scalisz gałąź z poprawką i wypchniesz zmiany na serwer produkcyjny.
- Przełączysz się na powrót do gałęzi z nową funkcją i będziesz kontynuować pracę.

Krótka i prosta historia zmian Twojego projektu



Pracujesz na problemem #53

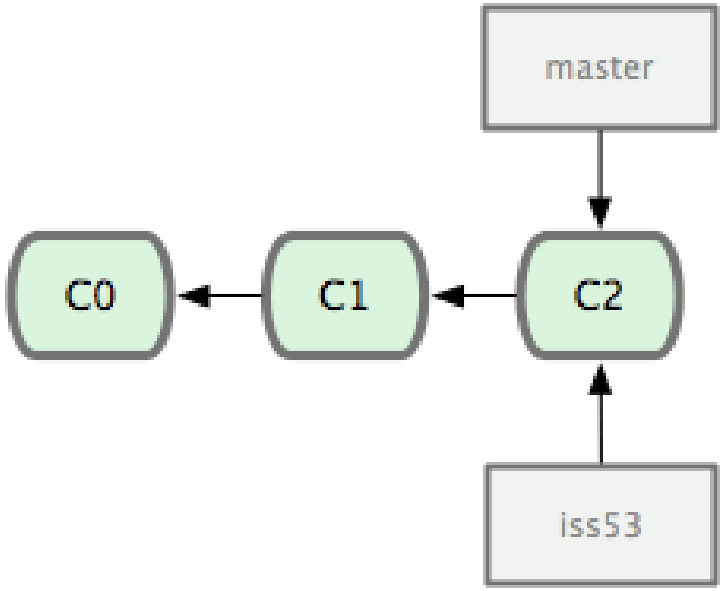
- utworzysz nową gałąź by się zając problemem #53

$ git checkout -b iss53

Switched to a new branch "iss53”

- Lub

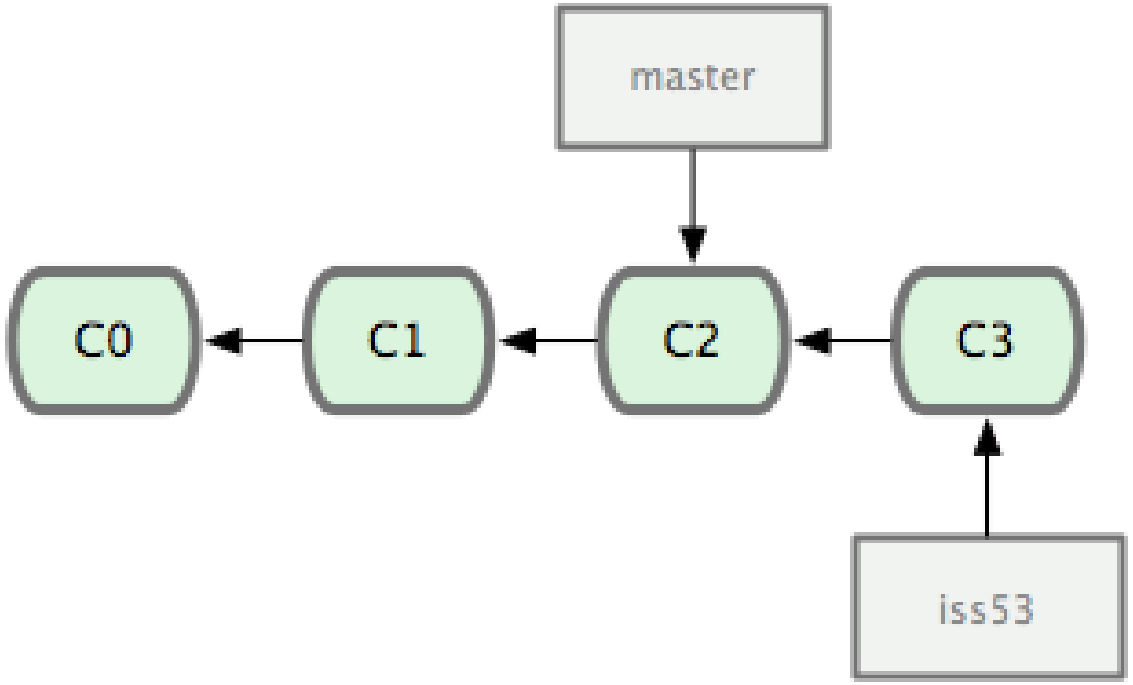
$ git branch iss53 $ git checkout iss53



Kolejne zmiany

$ vim index.html

$ git commit -a -m 'nowa stopka [#53]'



Potrzeba powrotu do podstawowej gałęzi

$ git checkout master

Switched to branch "master„

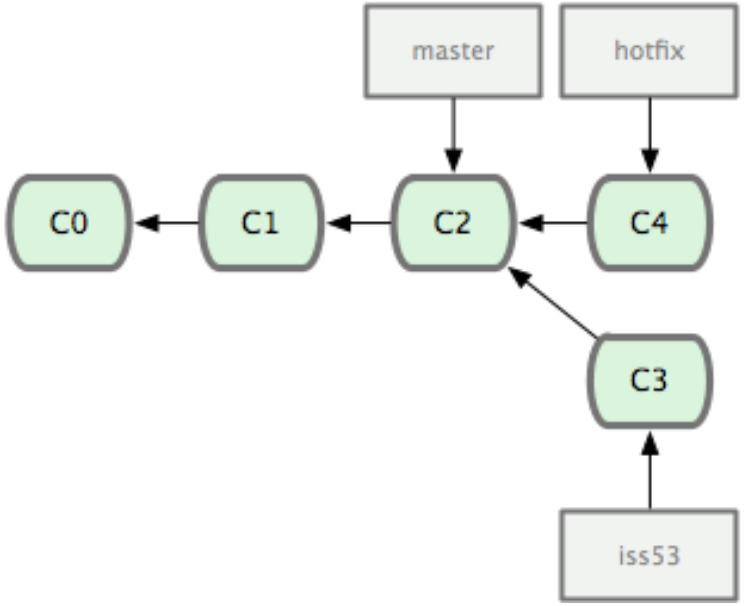
$ git checkout -b 'hotfix' Switched to a new branch "hotfix"

$ vim index.html

$ git commit -a -m 'poprawiony adres e-mail'

[hotfix]: created 3a0874c: "poprawiony adres e-mail"

1 files changed, 0 insertions(+), 1 deletions(-)



Scalanie

$ git checkout master

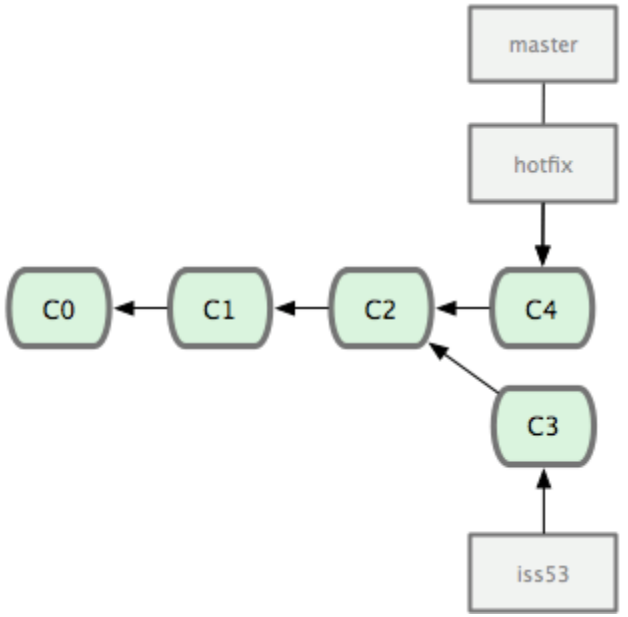
$ git merge hotfix

Updating f42c576..3a0874c

Fast forward

README | 1 –

1 files changed, 0 insertions(+), 1 deletions(-)



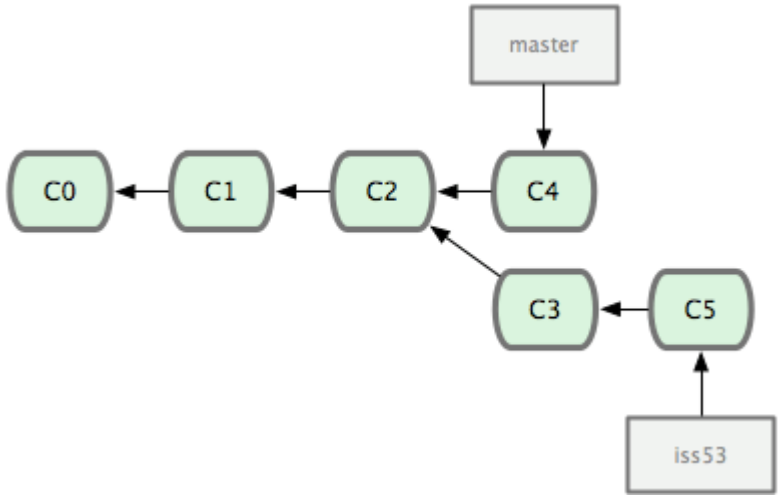
„Fast forward” gdy zmian wskazywany przez scalaną gałąź był bezpośrednim rodzicem aktualnego zestawu zmian, Git przesuwa wskaźnik do przodu.

Po prawidłowym scaleniu można usunąć gałąź

$ git branch -d hotfix

Deleted branch hotfix (3a0874c).

Wracamy do pracy nad #53



$ git checkout iss53

Switched to branch "iss53"

$ vim index.html

$ git commit -a -m 'skończona nowa stopka [#53]'

[iss53]: created ad82d7a: "skończona nowa stopka [#53]"

1 files changed, 1 insertions(+), 0 deletions(-)

Praca, jaka została wykonana na gałęzi hotfix nie jest uwzględniona w plikach w gałęzi iss53. Można scalić zmiany z gałęzi master do gałęzi iss53, uruchamiając git merge master

Scalenie

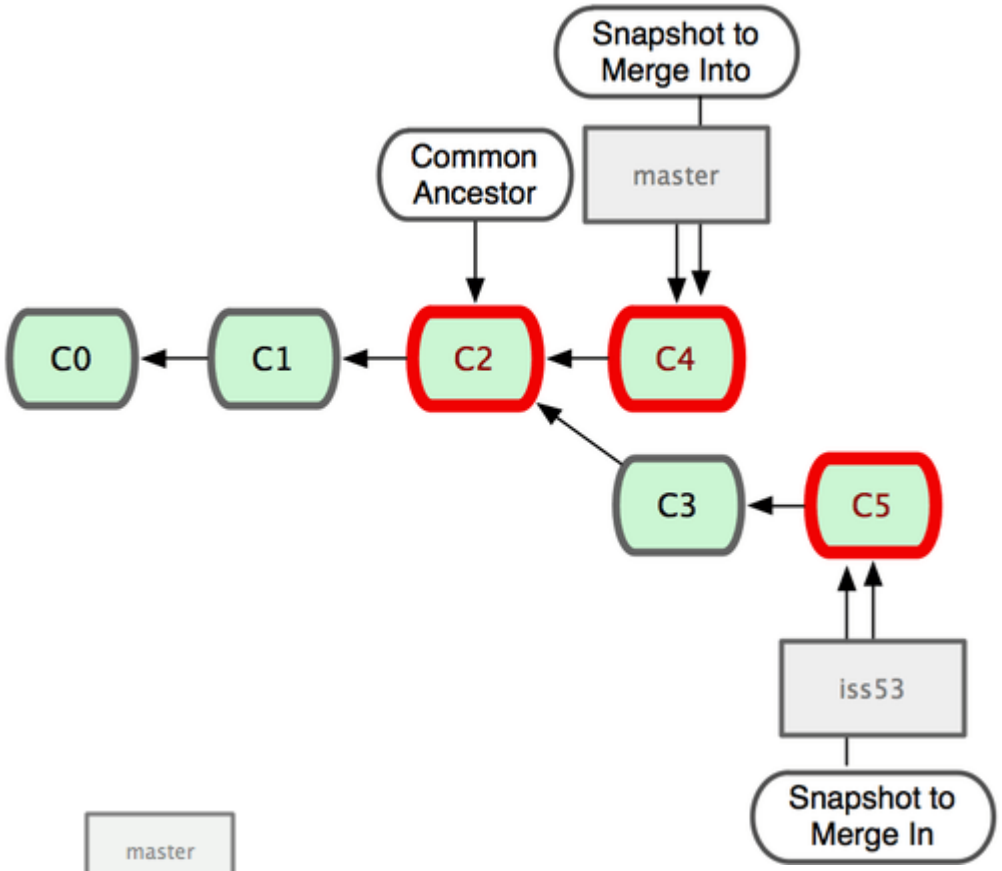
$ git checkout master

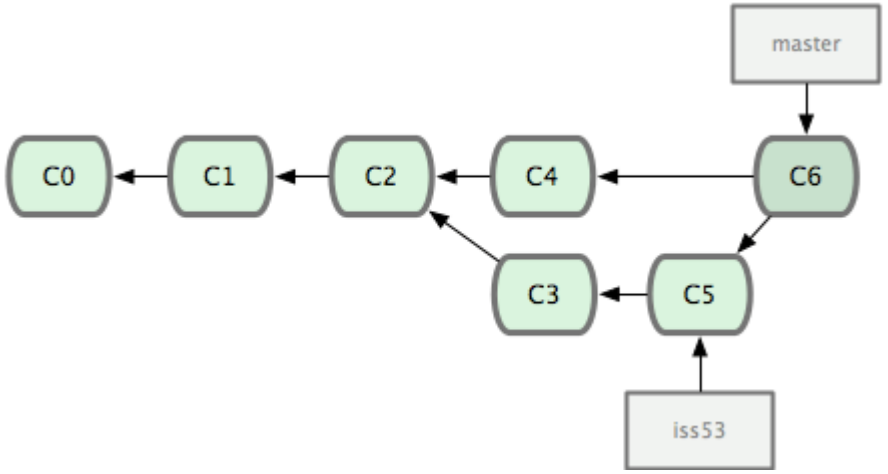
$ git merge iss53

Merge made by recursive.

README | 1 +

1 files changed, 1 insertions(+), 0 deletions(-)





Konflikty scalania

- Od czasu do czasu proces scalania nie przebiega gładko
- Ten sam plik zmieniony w różny sposób w obu scalanych gałęziach, Git nie będzie w stanie scalić ich samodzielnie

$ git merge iss53

Auto-merging index.html

CONFLICT (content): Merge the conflict in index.html

Automatic merge failed; fix conflicts and then commit result.

Konflikty scalania

```
<<<<<<< HEAD:index.html
<div id="footer">contact : email.support@github.com</div>
=======
<div id="footer">
  please contact us at support@github.com
</div>
>>>>>>> iss53:index.html
```

- Aby rozwiązać konflikt, musisz wybrać jedną lub druga wersję albo własnoręcznie połączyć zawartość obu.

** **

please contact us at email.support@github.com

** **

- **git mergetool** – graficzne narzędzie do rozwiazywania problemów

Zarządzanie gałęziami

$ git branch

iss53

*** master**

testing

znak *, którym poprzedzona została gałąź master: wskazuje on aktywną gałąź. Oznacza to, że jeżeli w tym momencie zatwierdzisz zmiany, wskaźnik gałęzi master zostanie przesunięty do przodu wraz z nowo zatwierdzonymi zmianami.

- Aby obejrzeć ostatni zatwierdzony zestaw zmian na każdej z gałęzi, możesz użyć polecenia git branch -v:

$ git branch -v

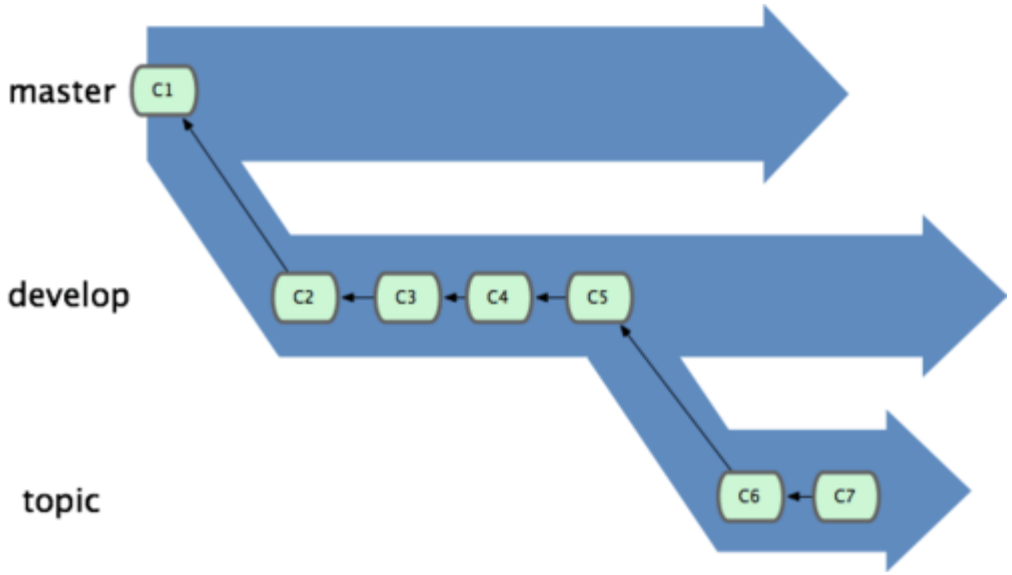
iss53 93b412c fix javascript issue

*** master 7a98805 Merge branch 'iss53'**

testing 782fd34 add scott to the author list in the readmes

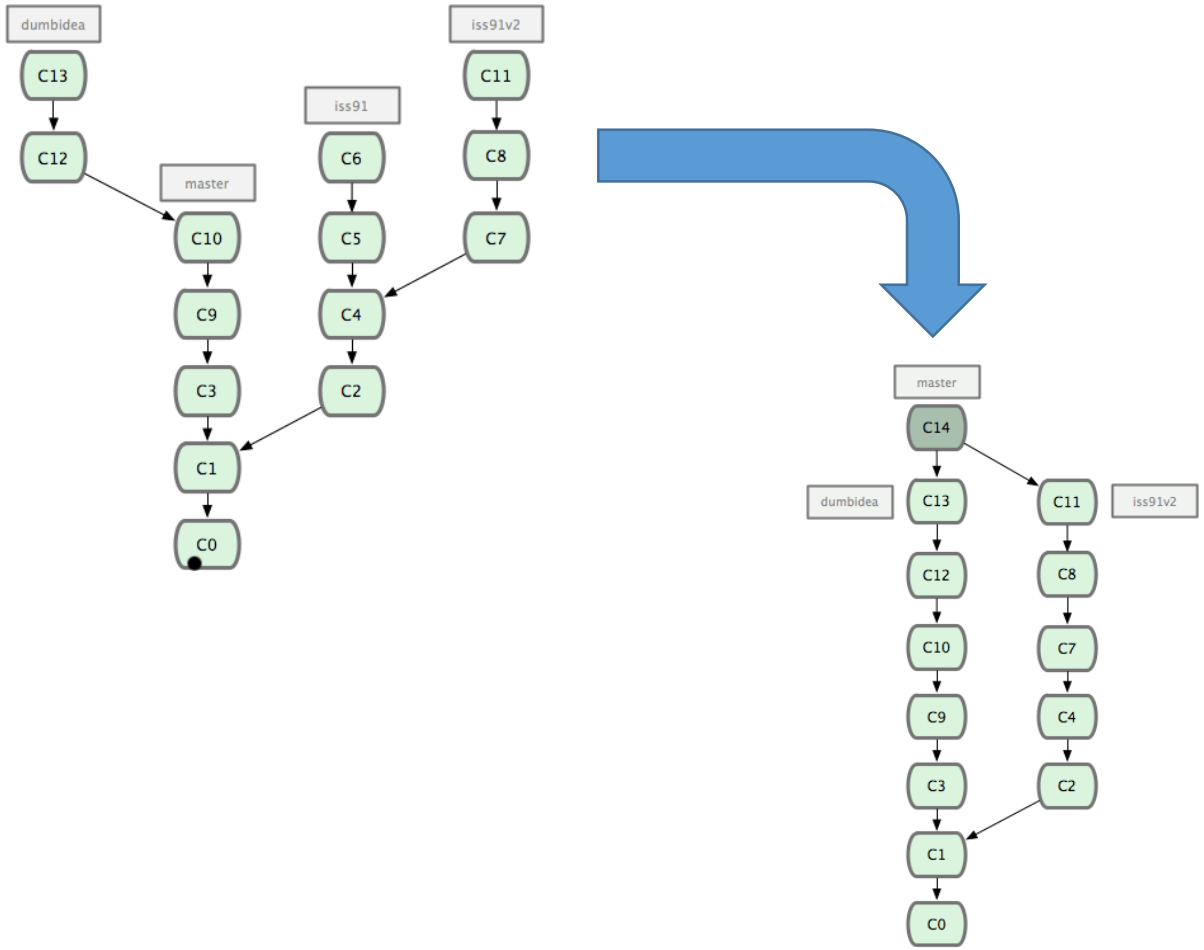
Sposoby pracy z gałęziami

- Gałęzie długodystansowe
- można utrzymywać kilka gałęzi, które są zawsze otwarte i których używa się dla różnych faz w cyklu rozwoju;
- można scalać zmiany regularnie z jednych gałęzi do innych.
- **master** jedynie stabilny kod — możliwe, że jedynie kod, który już został albo w najbliższej przyszłości zostanie wydany. Równolegle utrzymywane są inna gałąź o nazwie **develop** lub **next** — zawarta w niej praca nie musi być zawsze stabilna, lecz po stabilizacji może być scalona do gałęzi master.



Sposoby pracy z gałęziami

- Gałęzie tematyczne
- Przydadzą się w każdym projekcie, niezależnie od jego rozmiarów.
- Gałąź tematyczna to gałąź krótkodystansowa, którą tworzysz i używasz w celu stworzenia pojedynczej funkcji lub innych tego rodzaju zmian.



Praca ze zdalnym repozytorium

- współpracować za pośrednictwem Gita z innymi ludźmi
- Zdalne repozytorium to wersja twojego projektu utrzymywana na serwerze dostępnym poprzez Internet lub inną sieć
- Współpraca w grupie zakłada zarządzanie zdalnymi repozytoriami oraz wypychanie zmian na zewnątrz i pobieranie ich

Wyświetlanie zdalnych repozytoriów

```
$ git clone git://github.com/schacon/ticgit.git
Initialized empty Git repository in /private/tmp/ticgit/.git/
remote: Counting objects: 595, done.
remote: Compressing objects: 100% (269/269), done.
remote: Total 595 (delta 255), reused 589 (delta 253)
Receiving objects: 100% (595/595), 73.31 KiB | 1 KiB/s, done.
Resolving deltas: 100% (255/255), done.
$ cd ticgit
$ git remote
origin
```

Wyświetlanie zdalnych repozytoriów

- Dodanie parametru -v spowoduje dodatkowo wyświetlenie przypisanego do skrótu, pełnego, zapamiętanego przez Gita, adresu URL:

```
$ git remote –v
origin git://github.com/schacon/ticgit.git
```

- Więcej niż jedno zdalne repozytorium

```
$ git remote -v
bakkdoor  git://github.com/bakkdoor/grit.git
cho45     git://github.com/cho45/grit.git
defunkt   git://github.com/defunkt/grit.git
koke      git://github.com/koke/grit.git
origin    git@github.com:mojombo/grit.git
```

Dodawanie zdalnych repozytoriów

git remote add [skrót] [url]

```
$ git remote
origin
$ git remote add pb
git://github.com/paulboone/ticgit.git
$ git remote -v
origin  git://github.com/schacon/ticgit.git
pb  git://github.com/paulboone/ticgit.git
```

- Skrót daje możliwość posługiwania się nim zamiast adresem url

Pobieranie i wciąganie zmian ze zdalnych repozytoriów

`$ git fetch [nazwa-zdalengo-repozytorium]`

- sięga do zdalnego projektu i pobiera z niego wszystkie dane, których jeszcze nie masz
- `git fetch origin`
- pobierze każdą nową pracę jaka została wypchnięta na oryginalny serwer od momentu sklonowania go przez ciebie (lub ostatniego pobrania zmian)
- fetch pobiera dane do lokalnego repozytorium - **nie scala jednak automatycznie zmian** z żadnym z twoich plików roboczych jak i w żaden inny sposób tych plików nie modyfikuje

Pobieranie i wciąganie zmian ze zdalnych repozytoriów

`git pull`

- Pozwala automatycznie pobrać dane (fetch) i je scalić (merge) z lokalnymi plikami

Wypychanie zmian na zewnątrz

`git push [nazwa-zdalnego-repo] [nazwa-gałęzi]`

- Wypychanie na gałąź główną **master** na oryginalny serwer źródłowy **origin** (**ponownie, klonowanie ustawia obie te nazwy - master i origin - domyślnie i automatycznie**)
- Polecenie zadziała tylko jeśli:
  - sklonowałeś repozytorium z serwera do którego masz prawo zapisu
  - nikt inny w międzyczasie nie wypchnął własnych zmian
    - najpierw zespolić (pobrać i scalić) najnowsze zmiany ze zdalnego repozytorium zanim będziesz mógł wypchnąć własne.

Inspekcja zdalnych zmian

```
git remote show [nazwa-zdalnego-repo]
$ git remote show origin
* remote origin
  URL: git://github.com/schacon/ticgit.git
  Remote branch merged with 'git pull' while on branch
master
    master
  Tracked remote branches
    master
    ticgit
```

Usuwanie i zmiana nazwy zdalnych repozytoriów

- zmienić nazwę odnośnika

```
$ git remote rename pb paul
$ git remote
origin
Paul
```

- Usuwanie

```
$ git remote rm paul
$ git remote
origin
```

Gałęzie zdalne

- są odnośnikami do stanu gałęzi w zdalnym repozytorium
- są to lokalne gałęzie, których nie można zmieniać; są one modyfikowane automatycznie za każdym razem, kiedy wykonujesz jakieś operacje zdalne
- przybierają następującą formę

`(nazwa zdalnego repozytorium)/(nazwa gałęzi)`

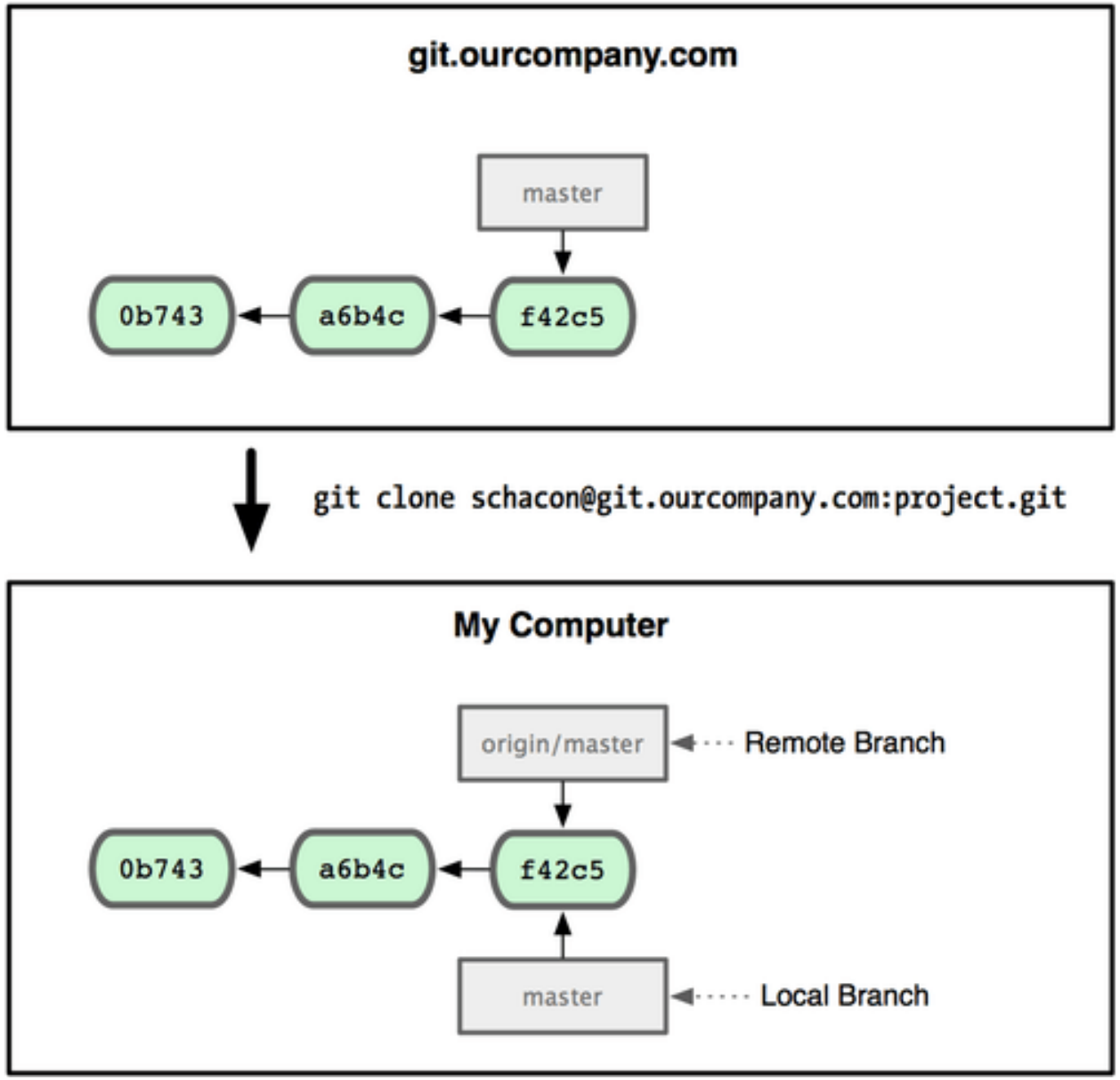
- Przykład:

`origin/master`

`origin/iss53`

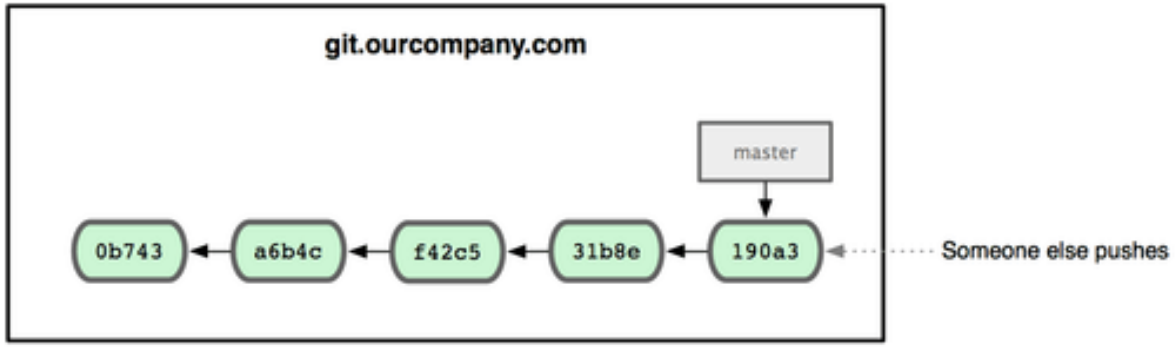
Gałęzie zdalne

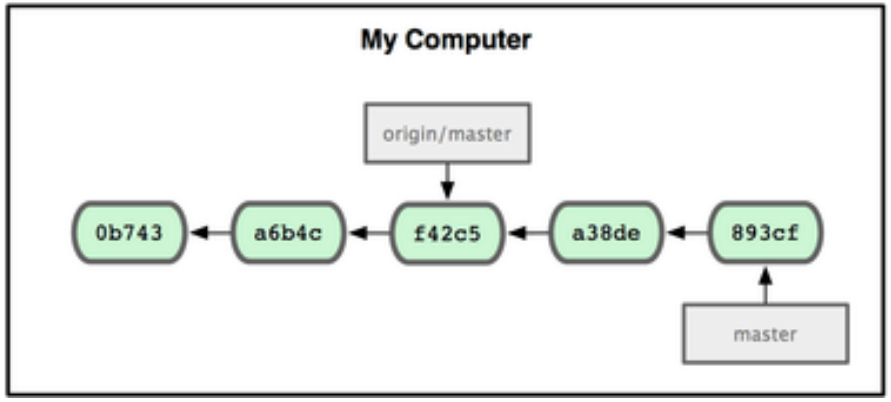
Po klonowaniu zdalnego repozytorium
- Automatycznie nazwie je jako origin
- Ustawi wskaźnik do gałęzi master
- Nazwie go origin/master
  - bez możliwości przesuwania
- Powstanie także lokalna master do pracy



Gałęzie zdalne

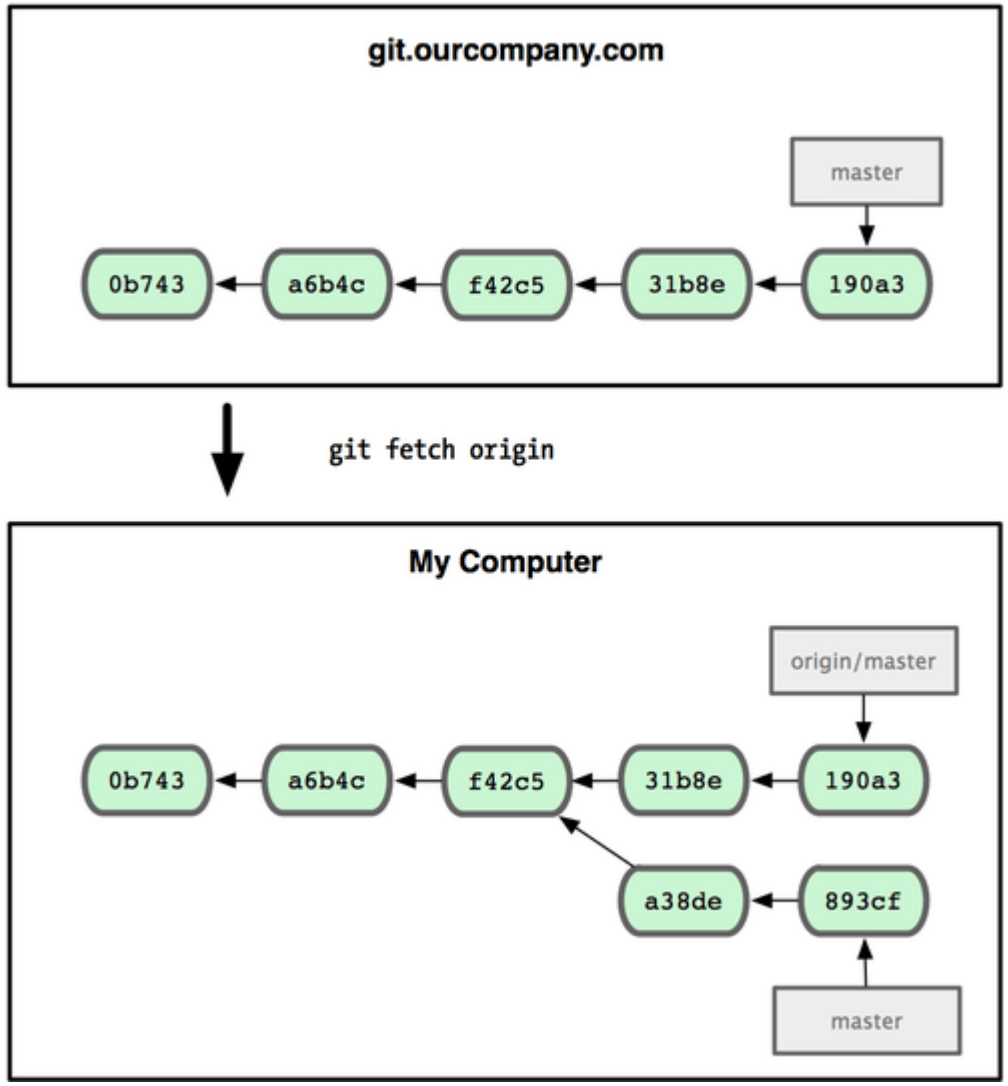
- wykonasz jakąś pracę na gałęzi głównej, a w międzyczasie ktoś inny wypchnie zmiany na git.ourcompany.com i zaktualizuje jego gałąź główną, wówczas wasze historie przesuną się do przodu w różny sposób.
- dopóki nie skontaktujesz się z serwerem zdalnym, Twój wskaźnik origin/master nie przesunie się





Gałęzie zdalne

- synchronizacja zmian
  - git fetch origin
- zaktualizuje Twoją lokalną bazę danych przesuwając jednocześnie wskaźnik origin/master do nowej pozycji



Wypychanie zmian

- Wypychanie zmian
  - na serwer na którym posiadasz prawa zapisu.
  - twoje lokalne gałęzie nie są automatycznie synchronizowane z serwerem
- musisz jawnie określić gałęzie, których zmianami chcesz się podzielić
- Jeśli posiadasz gałąź o nazwie serverfix, w której chcesz współpracować z innymi, możesz wypchnąć swoje zmiany w taki sam sposób jak wypychałeś je w przypadku pierwszej gałęzi master

```
git push (nazwa zdalnego repozytorium) (nazwa gałęzi)
```

Wypychanie zmian

```
$ git push origin serverfix
Counting objects: 20, done.
Compressing objects: 100% (14/14), done.
Writing objects: 100% (15/15), 1.74 KiB, done.
Total 15 (delta 5), reused 0 (delta 0)
To git@github.com:schacon/simplegit.git
 * [new branch]      serverfix -> serverfix
```

Weź moją lokalną gałąź serverfix i wypchnij zmiany, aktualizując zdalną gałąź serverfix

Wypychanie zmian

```
$ git fetch origin
remote: Counting objects: 20, done.
remote: Compressing objects: 100% (14/14), done.
remote: Total 15 (delta 5), reused 0 (delta 0)
Unpacking objects: 100% (15/15), done.
From git@github.com:schacon/simplegit
 * [new branch]      serverfix    -> origin/serverfix
```

- podczas pobierania ściągasz nową, zdalną gałąź, nie uzyskujesz automatycznie lokalnej, edytowalnej jej wersji

Wypychanie zmian

```
$ git checkout -b serverfix origin/serverfix
Branch serverfix set up to track remote branch
refs/remotes/origin/serverfix.
Switched to a new branch "serverfix„
```

- Utworzy gałąź lokalna możliwą do edycji

Gałęzie śledzące

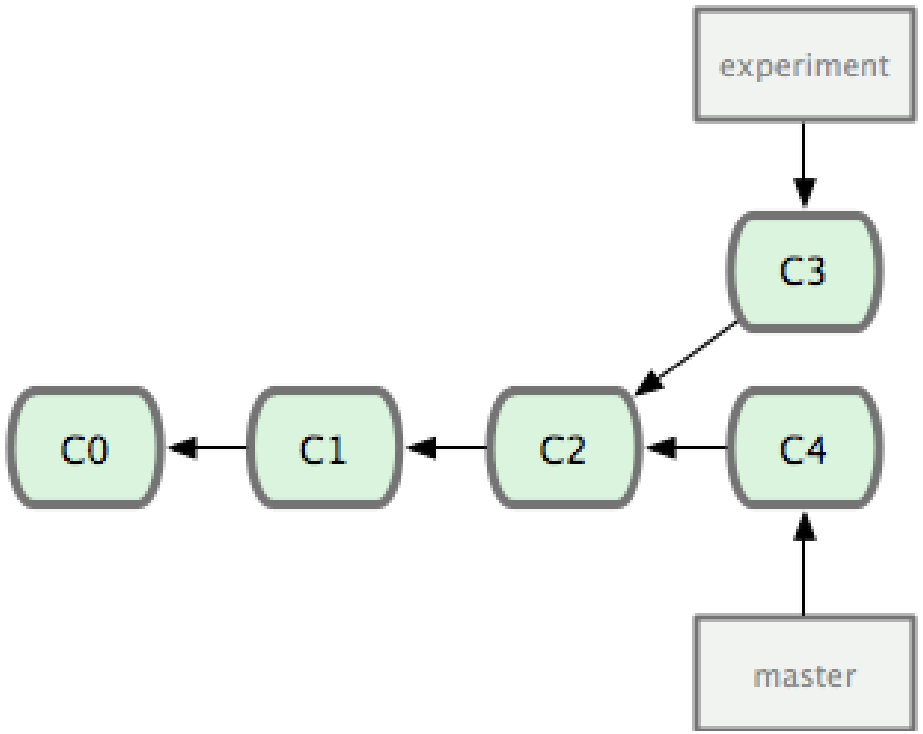
- Gałęzie śledzące są gałęziami lokalnymi, które posiadają bezpośrednią relację z gałęzią zdalną
- Po sklonowaniu repozytorium automatycznie tworzona jest gałąź master, która śledzi origin/master

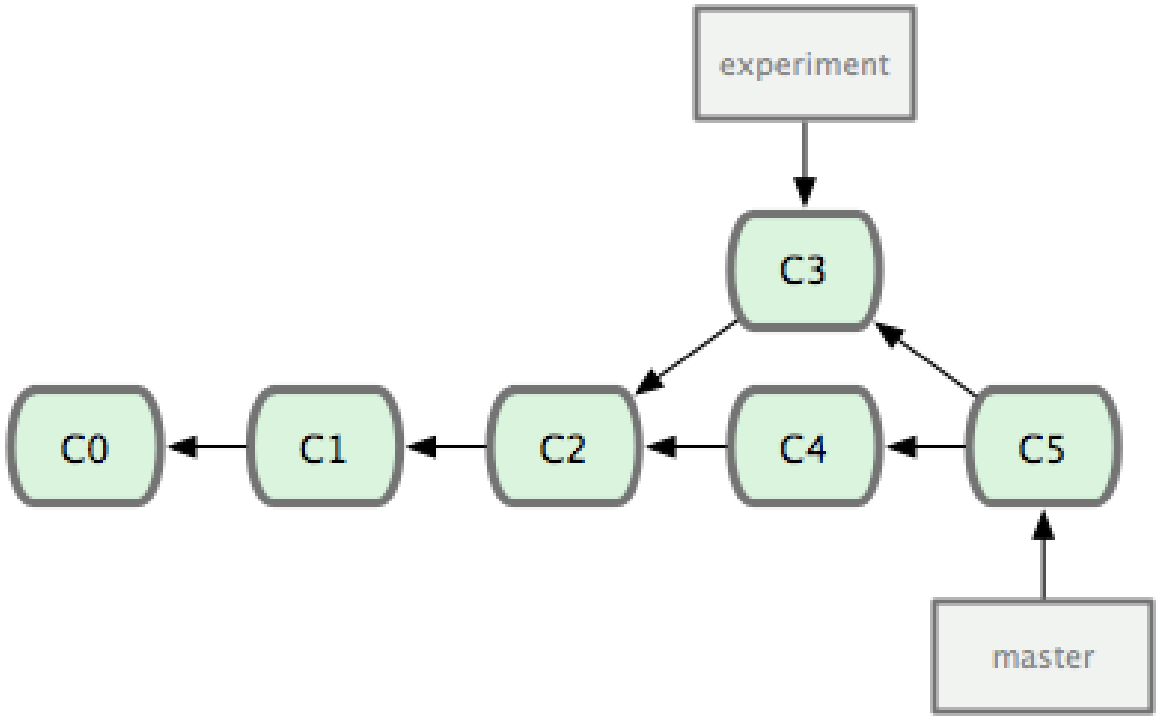
git checkout -b [gałąź] [nazwa zdalnego repozytorium]/[gałąź]

```
$ git checkout --track origin/serverfix
Branch serverfix set up to track remote branch
refs/remotes/origin/serverfix.
Switched to a new branch "serverfix"
```

Zmiana bazy

- Standardowy merge

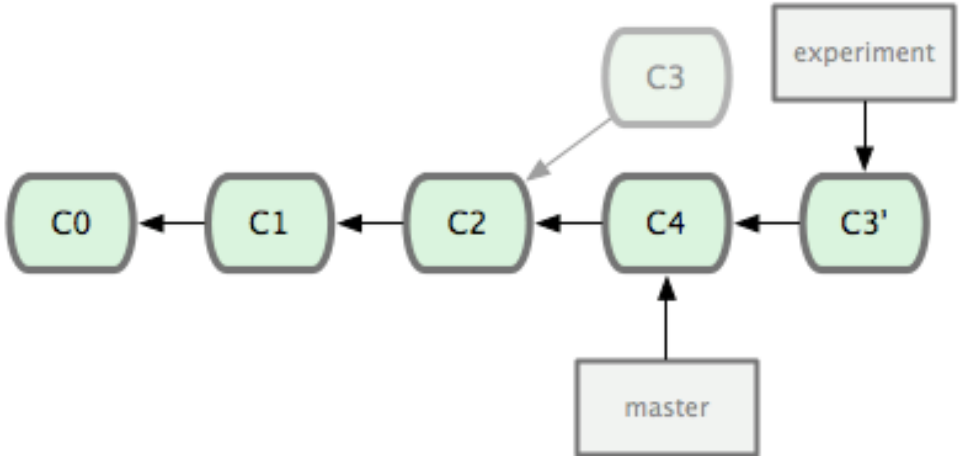




Zmiana bazy

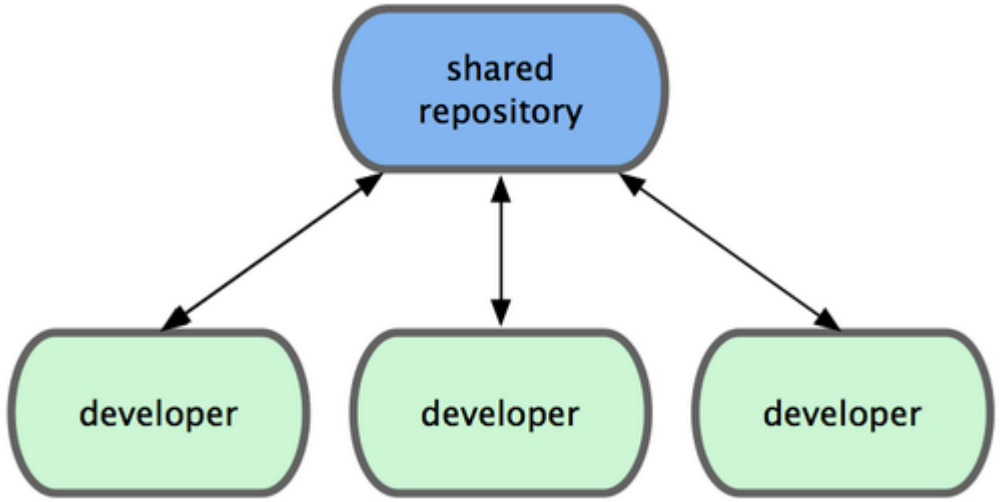
- rebase - możesz wziąć wszystkie zmiany, które zostały zatwierdzone w jednej gałęzi i zaaplikować je w innej

```
$ git checkout experiment
$ git rebase master
First, rewinding head to replay your work on top of it...
Applying: added staged command
```



Scentralizowany przepływ pracy

- Wszyscy mają uprawnienia do zapisu w gałęzi **master**
- Dwóch programistów – wgrywa zmiany:
  - Pierwszy to zrobi
  - Drugi musi pobrać zmiany, scalić i wgrać

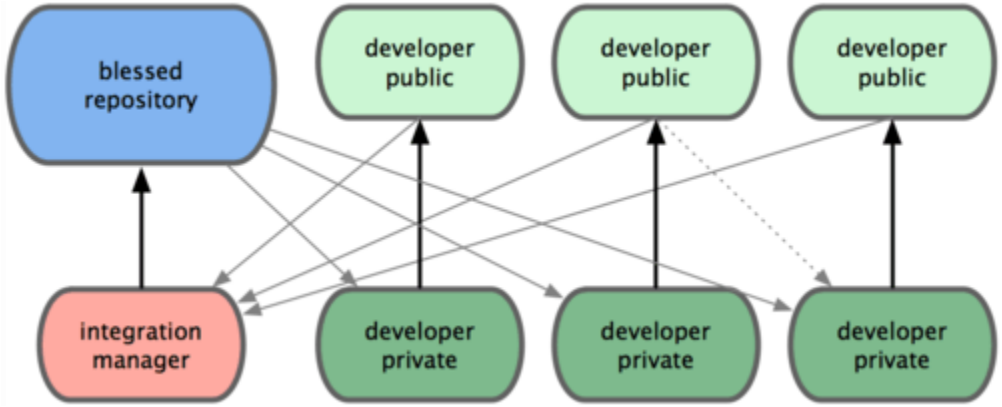


Przepływ pracy z osobą integrującą zmiany

- Każdy programista ma uprawnienia do zapisu do swojego własnego repozytorium oraz uprawnienia do odczytu do repozytorium innych osób w zespole
- Może istnieć jedno centralne - "oficjalne" repozytorium projektu

Przepływ pracy z osobą integrującą zmiany

1. Opiekun projektu wgrywa zmiany do publicznego repozytorium.
2. Programiści klonują to repozytorium i wprowadzają zmiany.
3. Programista wgrywa zmiany do swojego publicznego repozytorium.
4. Programista wysyła prośbę do opiekuna projektu, aby pobrał zmiany z jego repozytorium.
5. Opiekun dodaje repozytorium programisty jako repozytorium zdalne i pobiera zmiany.
6. Opiekun wgrywa włączone zmiany do głównego repozytorium.

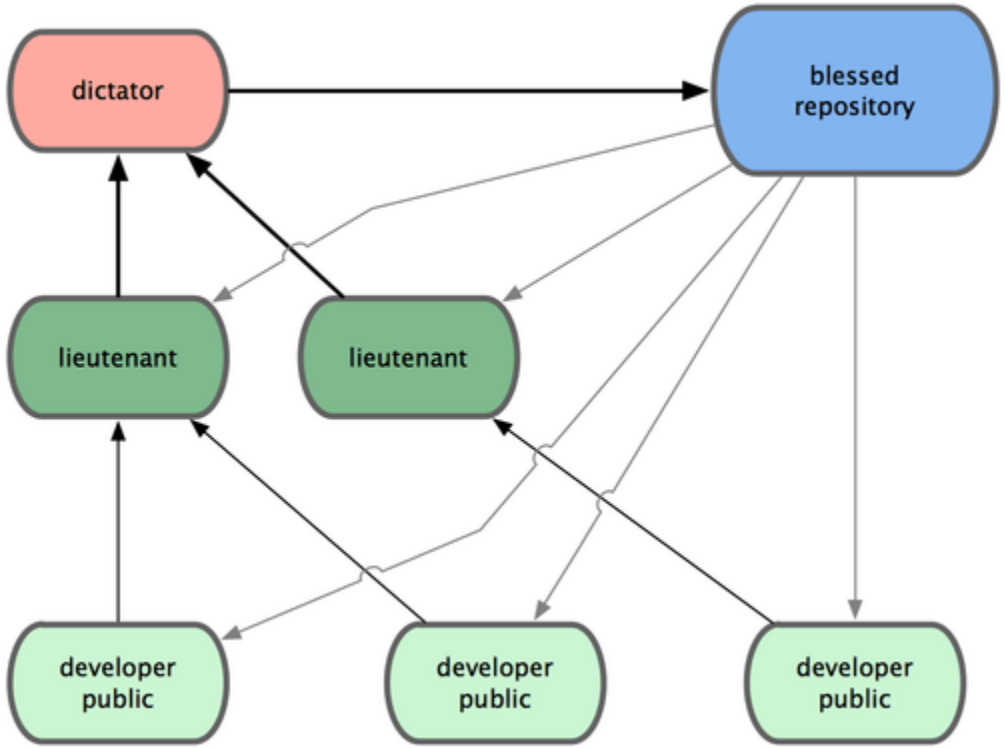


Przepływ pracy z dyktatorem i porucznikami

- Używany w bardzo dużych projektach
- Kilkoro opiekunów jest wydelegowanych do obsługi wydzielonych części repozytorium – porucznicy
- Jedna, główna osoba integrująca zmiany - miłościwy dyktator

Przepływ pracy z dyktatorem i porucznikami

1. Programiści pracują nad swoimi gałęziami tematycznymi, oraz wykonują "rebase" na gałęzi "master". Gałąź "master" jest tą pobraną od dyktatora.
2. Porucznicy włączają ("merge") zmiany programistów do swojej gałęzi "master".
3. Dyktator włącza ("merge") gałęzie "master" udostępnione przez poruczników do swojej gałęzi "master".
4. Dyktator wypycha ("push") swoją gałąź master do głównego repozytorium, tak aby inni programiści mogli na niej pracować.



use case - małe prywatne zespoły

- Repozytorium z jednym lub dwoma innymi współpracownikami
- Z zamkniętym kodem źródłowym - nie dostępnym do odczytu dla innych
- Deweloperzy mają uprawniania do wgrywania ("push") swoich zmian
- Przykład:
  - dwóch programistów (John i Jessica) pracuje z współdzielonym repozytorium

use case - małe prywatne zespoły

- John, klonuje repozytorium, wprowadza zmiany i zatwierdza je lokalnie

```
# Komputer Johna
$ git clone john@githost:simplegit.git
Initialized empty Git repository in
/home/john/simplegit/.git/
...
$ cd simplegit/
$ vim lib/simplegit.rb
$ git commit -am 'removed invalid default
value'
[master 738ee87] removed invalid default value
 1 files changed, 1 insertions(+), 1
deletions(-)
```

use case - małe prywatne zespoły

- Jessica, robi to samo — klonuje repozytorium i commituje zmianę

```
# Komputer Jessici
$ git clone jessica@githost:simplegit.git
Initialized empty Git repository in
/home/jessica/simplegit/.git/
...
$ cd simplegit/
$ vim TODO
$ git commit -am 'add reset task'
[master fbff5bc] add reset task
 1 files changed, 1 insertions(+), 0
deletions(-)
```

use case - małe prywatne zespoły

- Jessica wypycha swoje zmiany na serwer

```
# Komputer Jessici
$ git push origin master
...
To jessica@githost:simplegit.git
   1edee6b..fbff5bc  master -> master
```

use case - małe prywatne zespoły

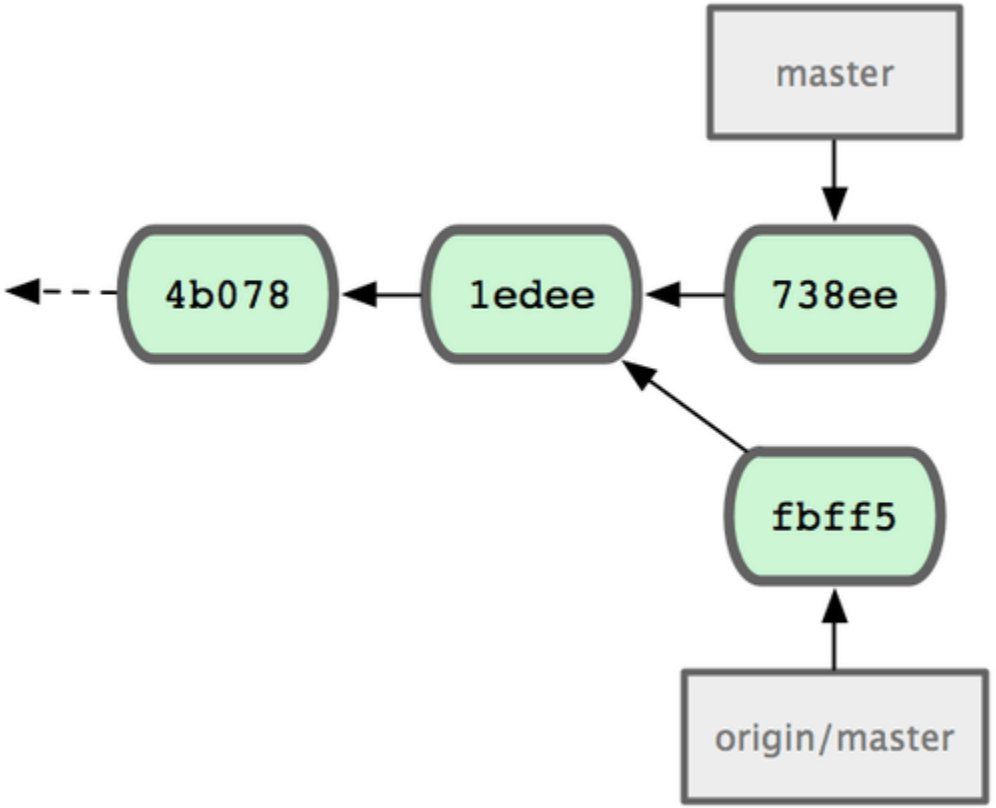
- John próbuje również wypchnąć swoje zmiany

```
# Komputer Johna
$ git push origin master
To john@githost:simplegit.git
 ! [rejected]        master -> master (non-fast forward)
error: failed to push some refs to 'john@githost:simplegit.git'
```

use case - małe prywatne zespoły

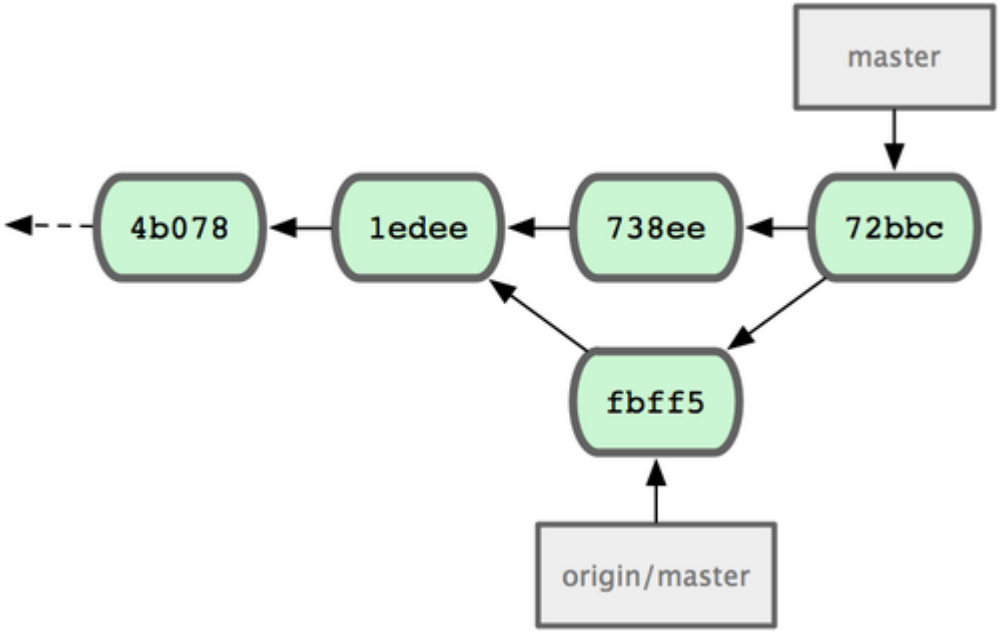
- musisz połączyć zmiany lokalnie
- John musi pobrać zmiany Jessici oraz włączyć je do swojego repozytorium zanim będzie wypychał swoje zmiany

```
$ git fetch origin
...
From john@githost:simplegit
 + 049d078...fbff5bc master     ->
origin/master
```

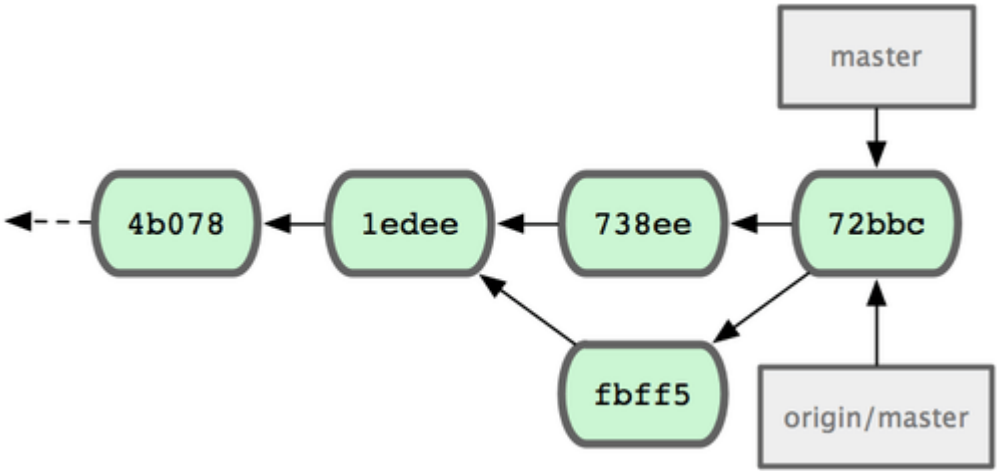


use case - małe prywatne zespoły

```
$ git merge origin/master
Merge made by recursive.
 TODO |    1 +
 1 files changed, 1 insertions(+), 0
deletions(-)
```

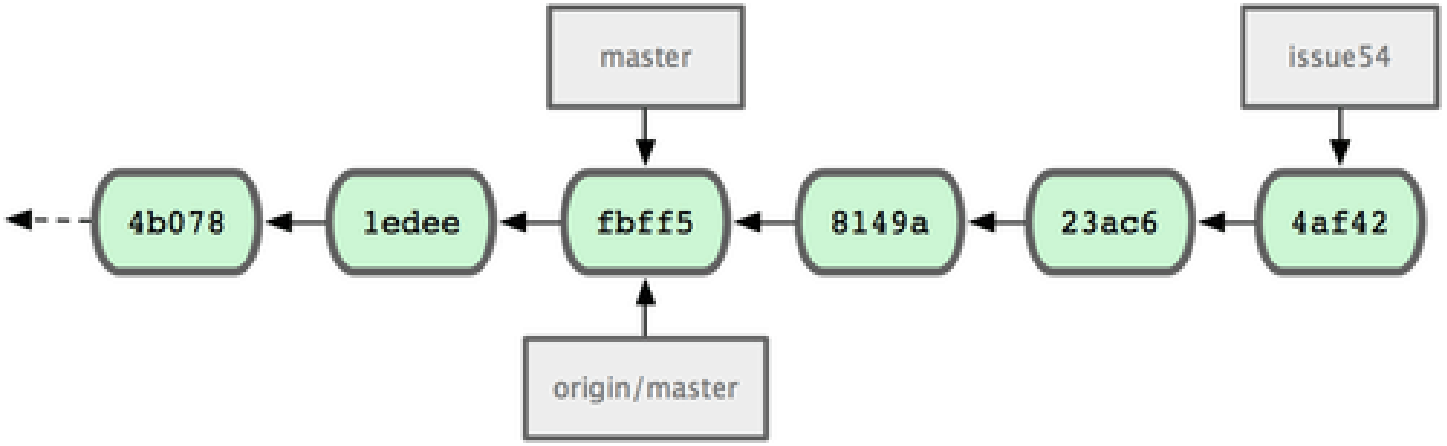



```
$ git push origin master
...
To john@githost:simplegit.git
   fbff5bc..72bbc59  master -> master
```



use case - małe prywatne zespoły

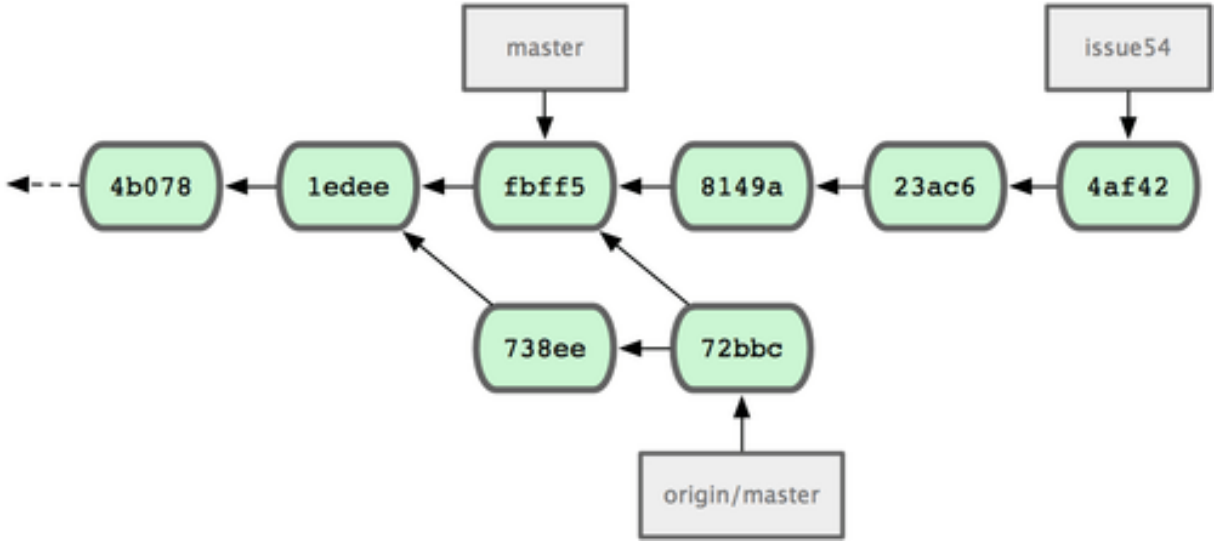
- Repozytorium Jessici:
  - pracowała na swojej tematycznej gałęzi
  - Stworzyła gałąź issue54
  - wprowadziła trzy zmiany w niej
  - nie pobrała jeszcze zmian Johna



use case - małe prywatne zespoły

- Jessica chce zsynchronizować się ze zmianami Johna

```
# Jessica's Machine
$ git fetch origin
...
From jessica@githost:simplegit
   fbff5bc..72bbc59  master     ->
origin/master
```



use case - małe prywatne zespoły

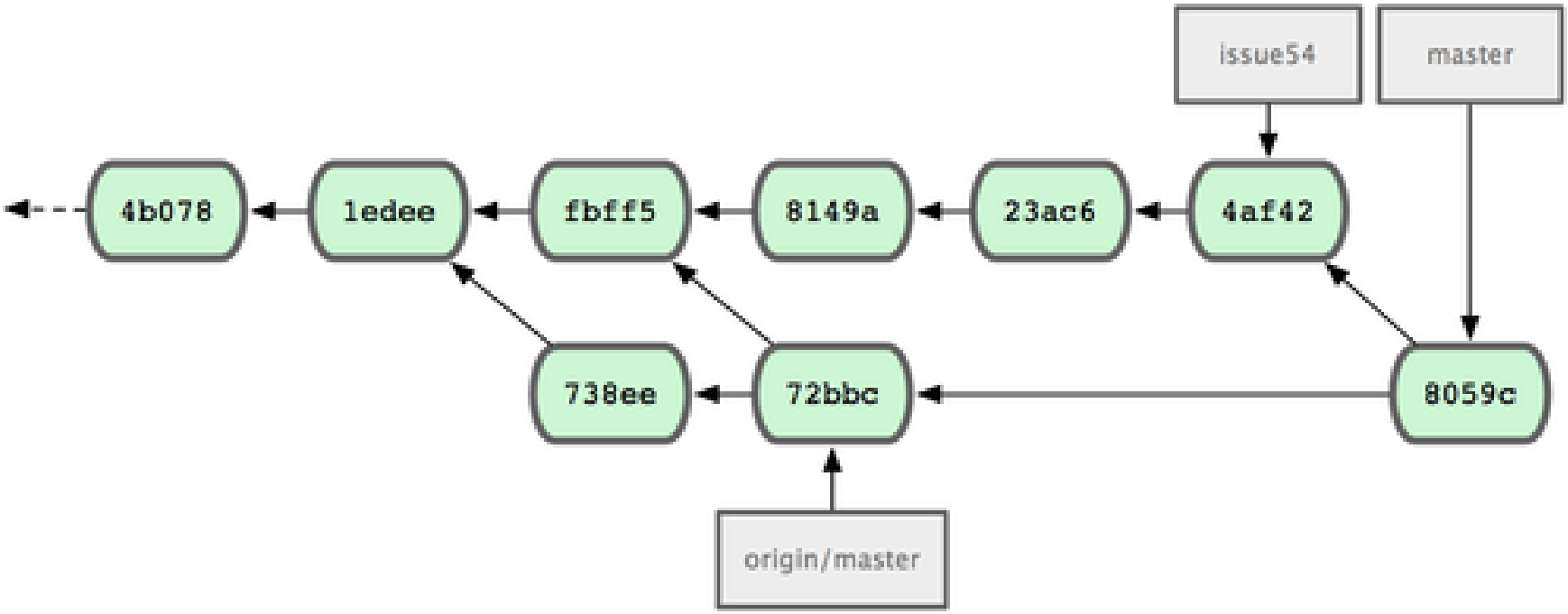
- Jessica może połączyć zmiany ze swojej gałęzi z gałęzią "master", włączyć zmiany Johna

```
$ git checkout master
Switched to branch "master"
Your branch is behind 'origin/master' by 2 commits, and can be fast-forwarded.
```

```
$ git merge issue54
Updating fbff5bc..4af4298
Fast forward
 README           |    1 +
 lib/simplegit.rb |    6 +++++-
 2 files changed, 6 insertions(+), 1
deletions(-)
```

```
$ git merge origin/master
Auto-merging lib/simplegit.rb
Merge made by recursive.
 lib/simplegit.rb |    2 +-
 1 files changed, 1 insertions(+), 1
deletions(-)
```

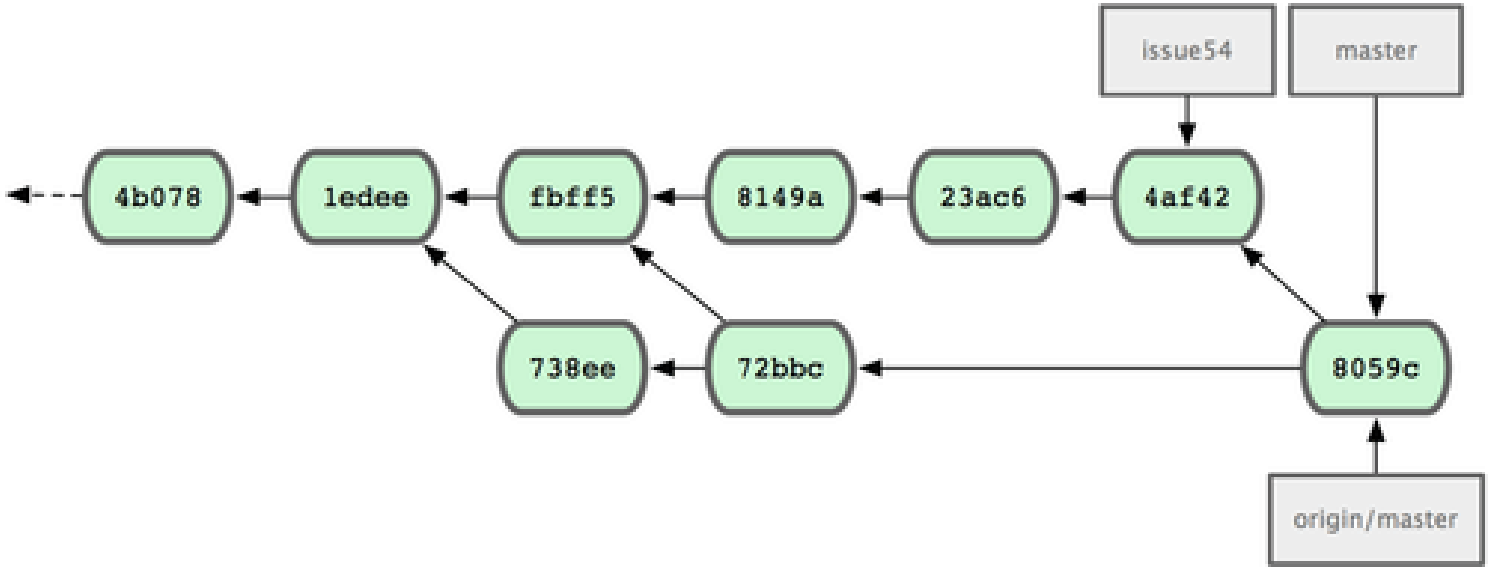
use case - małe prywatne zespoły



use case - małe prywatne zespoły

- Jesicca – wypycha zmiany

```
$ git push origin master
...
To jessica@githost:simplegit.git
   72bbc59..8059c15  master -> master
```



use case - małe prywatne zespoły