нетология

Ветвления в Git.

Андрей
Борю

Андрей Борю

Principal DevOps Engineer, Snapcart

andrey.borue | andrey.borue | andreyborue

План занятия

1. Ветвления и слияния
2. Удаленные ветки
3. Rebase (преобразование)
4. Хуки
5. Хранилища репозиториев
6. Итоги
7. Домашнее задание

Ветвления и слияния

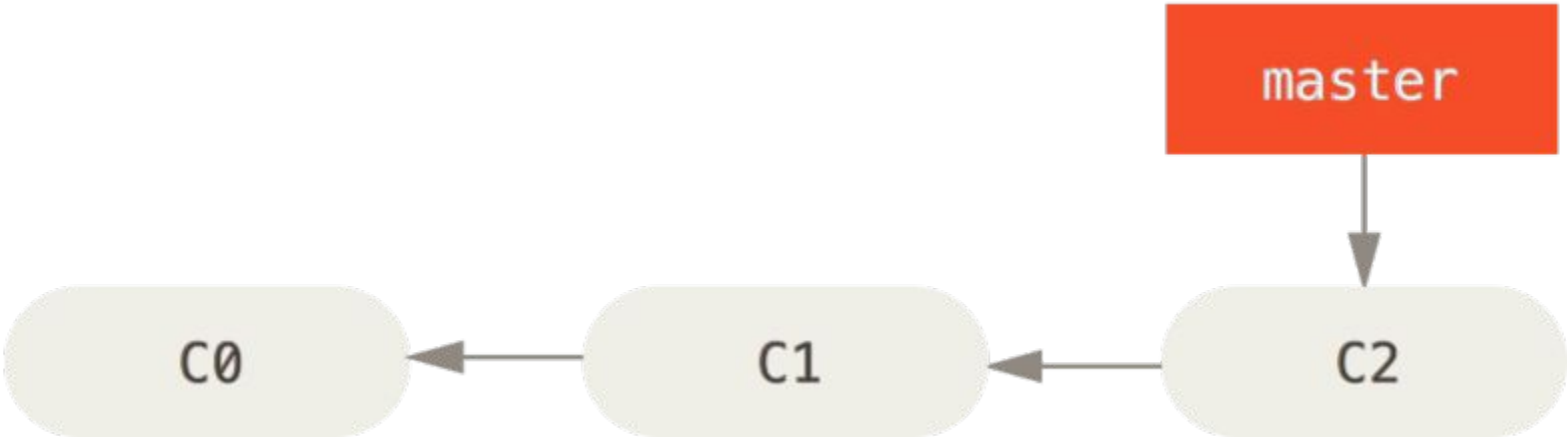
Схема работы над проектом

1. Вы работаете над проектом.
2. Создаете ветку для нового функционала (фичи) над которым будете работать.
3. Работаете в новой ветке не изменяя код в мастере.

Срочно необходимо исправить баг в основной ветке (мастере).

1. Переключаетесь в мастер.
2. Создаете ветку для исправления бага.
3. После исправления сливаете эту ветку в мастер.
4. Переключаетесь назад в ветку с фичей.

Простая история коммитов

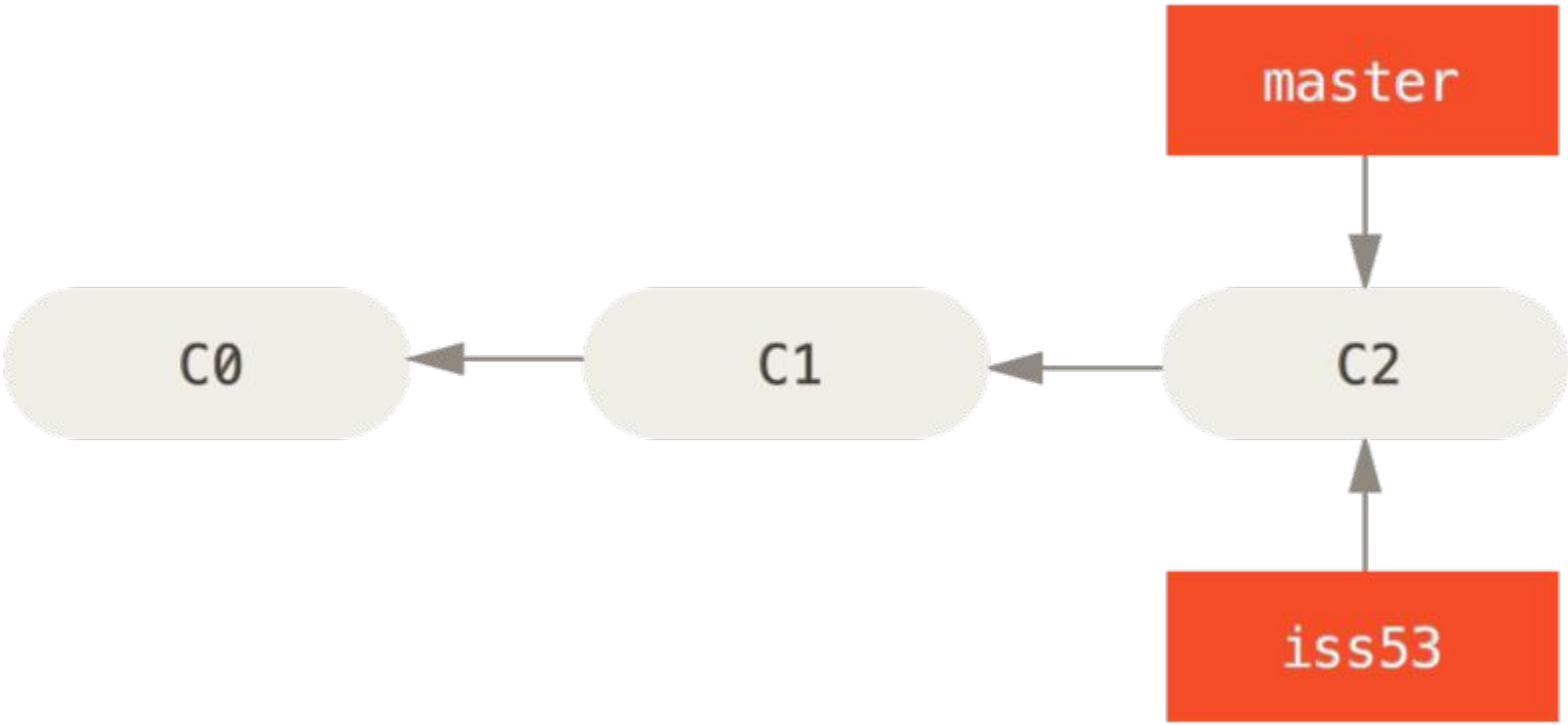


здесь и далее взято с сайта: git-scm.com

```
$ git checkout -b iss53
Switched to a new branch "iss53"
```

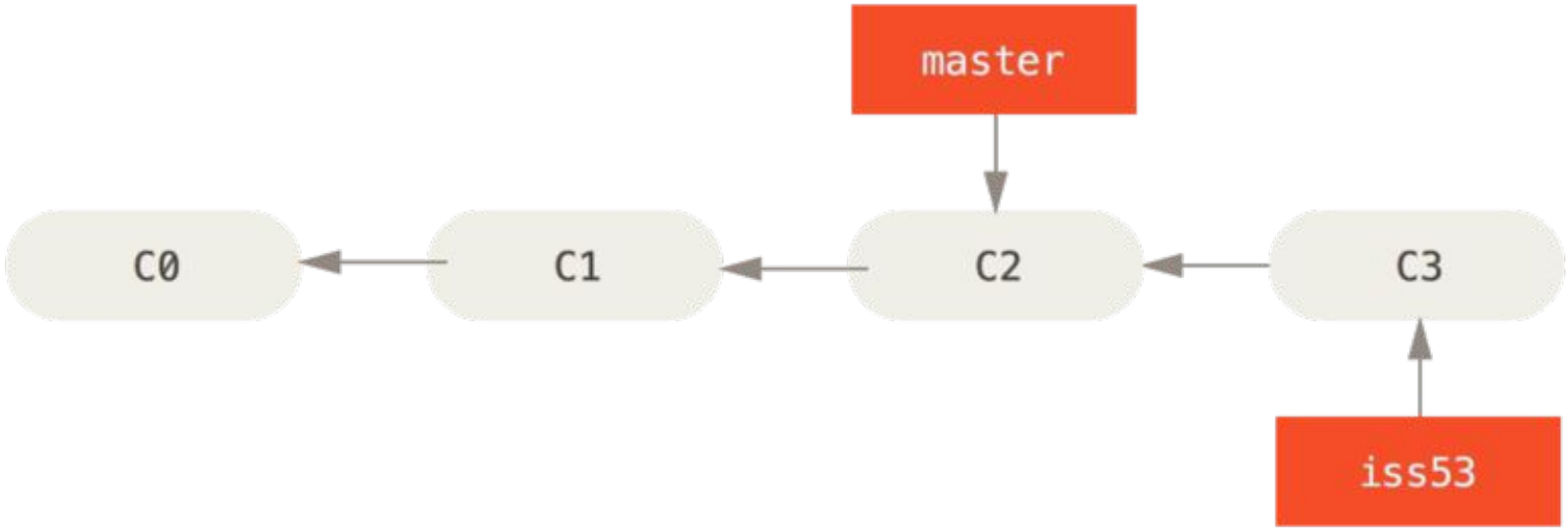
```
$ git branch iss53
$ git checkout iss53
```

Новый указатель ветки



Ветка двигается вперед

```
$ vim index.html
$ git commit -a -m 'added a new footer [issue 53]'
```



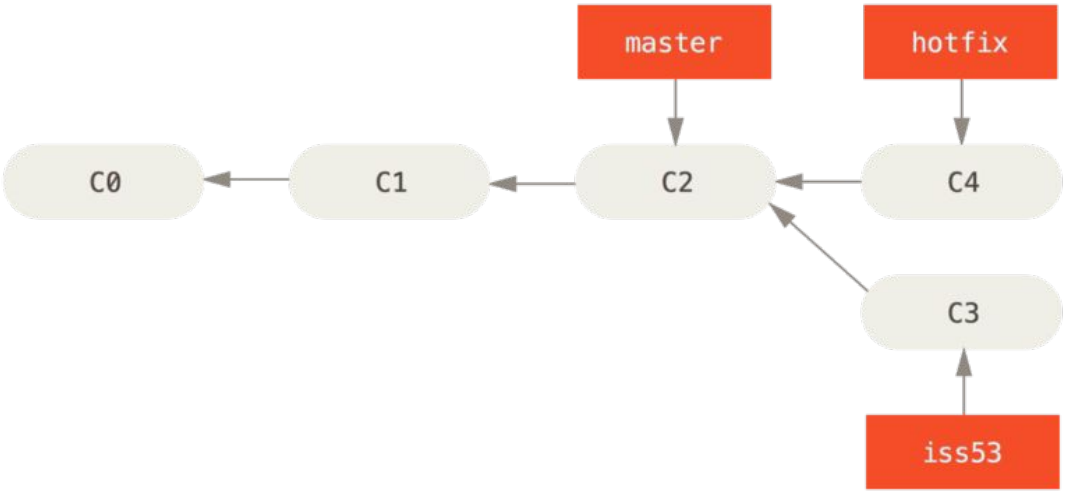
Необходим срочный хотфикс

$ git commit -m '[issue 53] ...'
$ git checkout master
Switched to branch 'master'

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

$ vim index.html

$ git commit -a -m 'fixed the broken email address'
[hotfix 1fb7853] fixed the broken email address
1 file changed, 2 insertions(+)



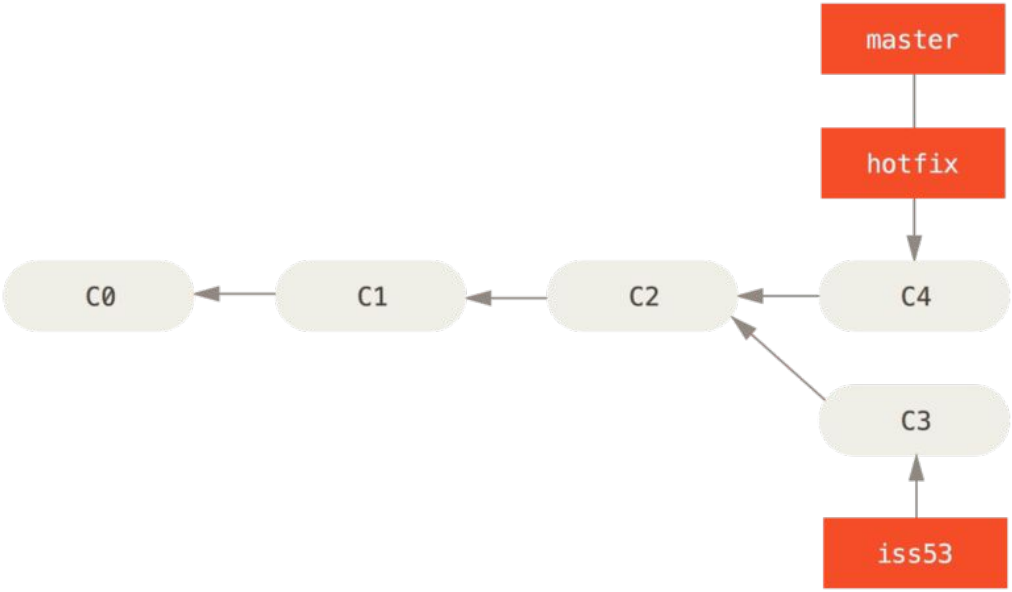
Мержим hotfix в master

$ git checkout master
$ git merge hotfix

```
Updating f42c576..3a0874c
Fast-forward
 index.html | 2 ++
 1 file changed, 2 insertions(+)
```

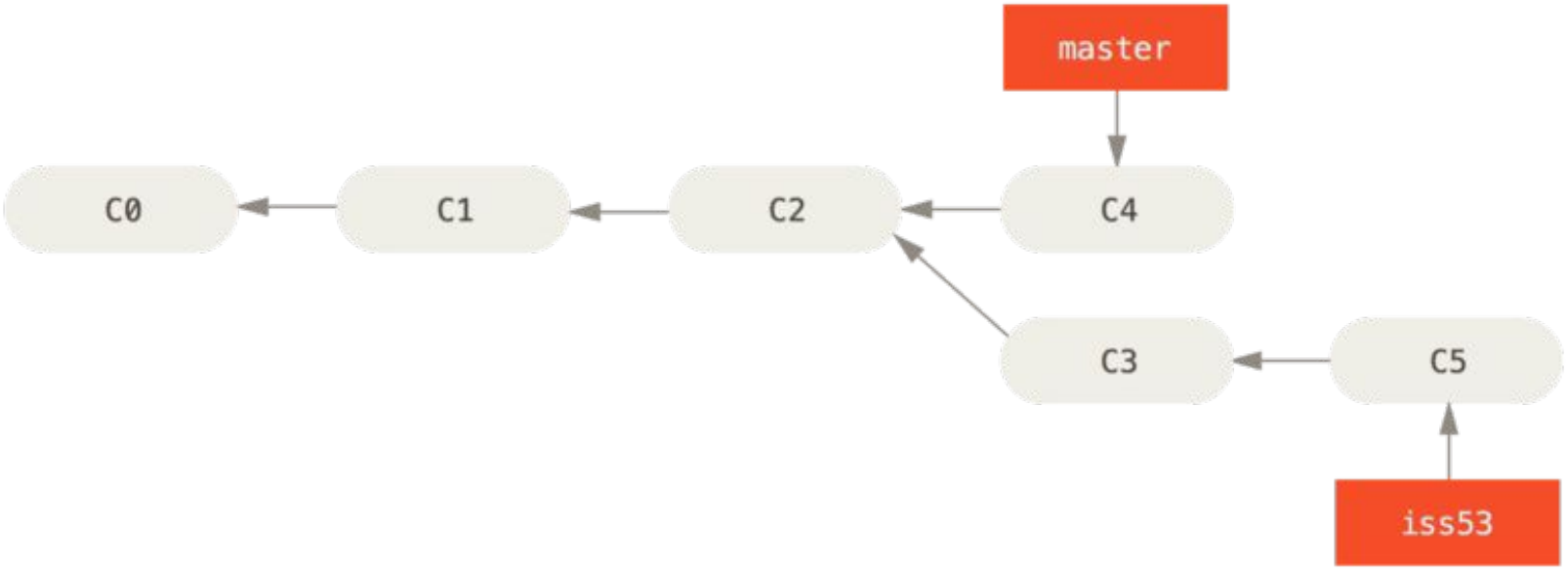
$ git branch -d hotfix

```
Deleted branch hotfix (3a0874c)
```



Продолжаем работать над задачей

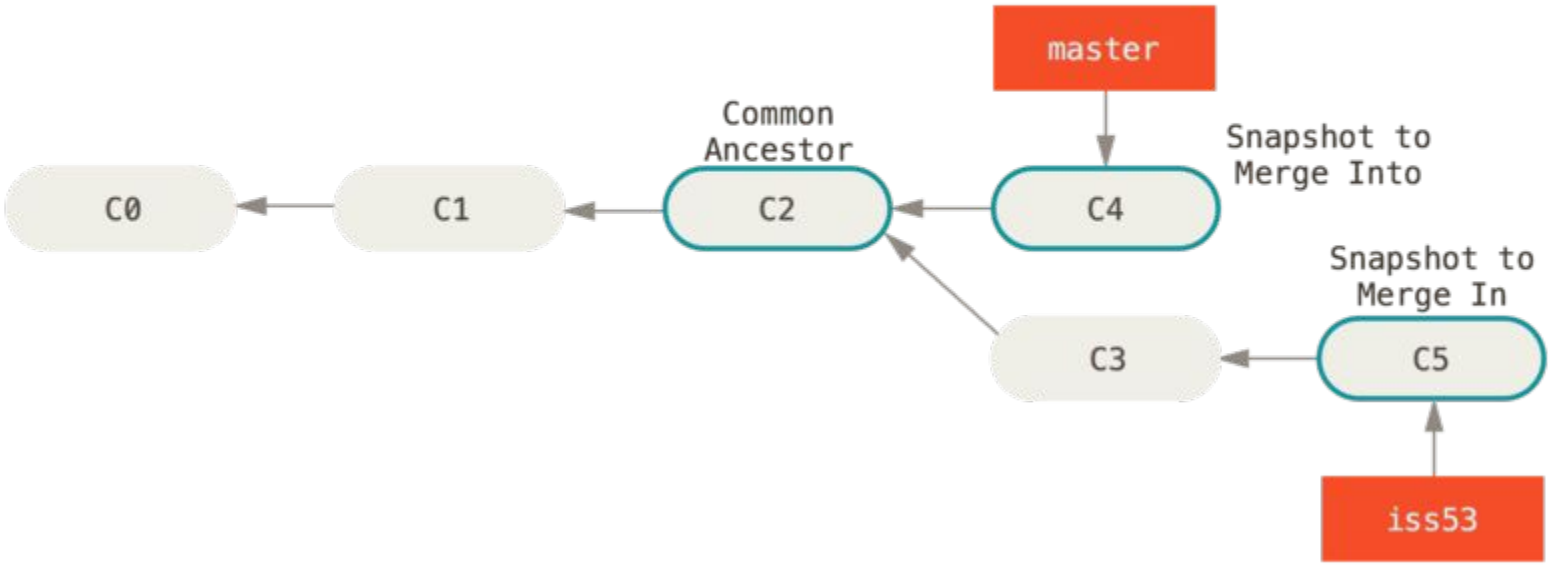
```
$ git checkout iss53
Switched to branch "iss53"

$ vim index.html
$ git commit -a -m 'finished the new footer [issue 53]'
[iss53 ad82d7a] finished the new footer [issue 53]
1 file changed, 1 insertion(+)
```



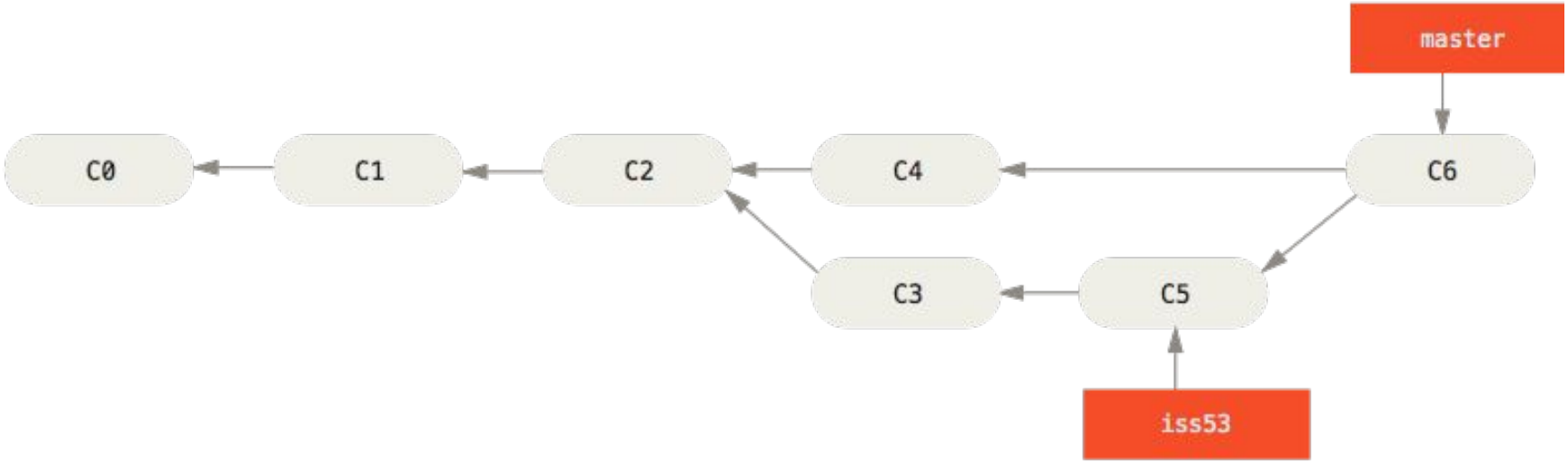
Мержим задачу в master

$ git checkout master
Switched to branch 'master'

$ git merge iss53
Merge made by the 'recursive' strategy.
index.html | 1 +
1 file changed, 1 insertion(+)



Создание мерж-коммита



$ git branch -d iss53

Deleted branch iss53 (4b9834b).

Конфликты

$ git merge iss53

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

$ 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:      index.html
no changes added to commit (use "git add" and/or "git commit -a")
```

Разрешение конфликта

$ vim index.html

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

```
<div id="footer">
 please contact us at support@netology.ru
</div>
```

Коммит после конфликта

```
$ git add index.html
$ git status

On branch master
All conflicts fixed but you are still merging.
  (use "git commit" to conclude merge)
Changes to be committed:
    modified:   index.html

$ git commit
```

```
Merge branch 'iss53'
Conflicts:
    index.html
#
# It looks like you may be committing a merge.
# If this is not correct, please remove the file
#         .git/MERGE_HEAD
# and try again.

...
```

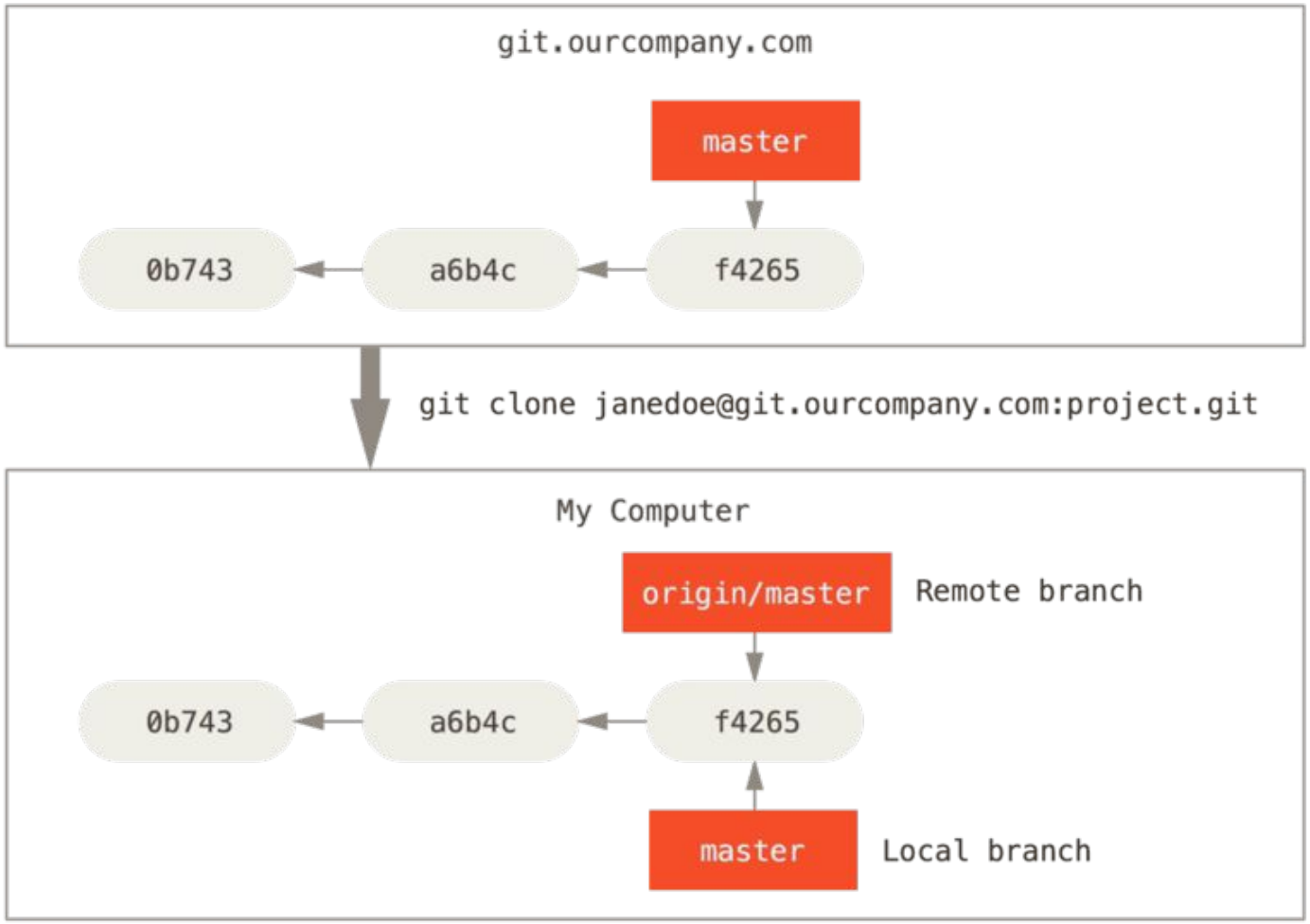
Удалённые ветки

Список удаленных веток

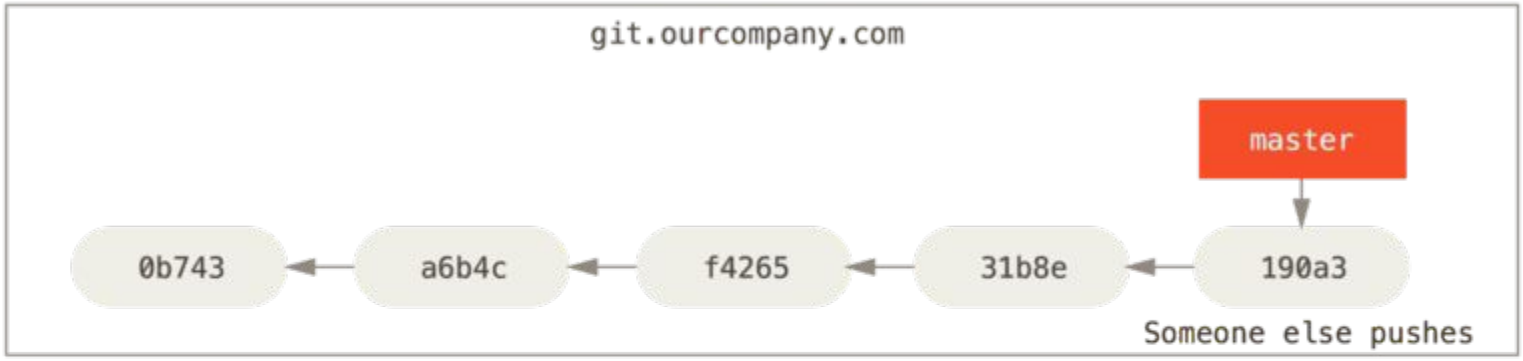
$ git remote show origin

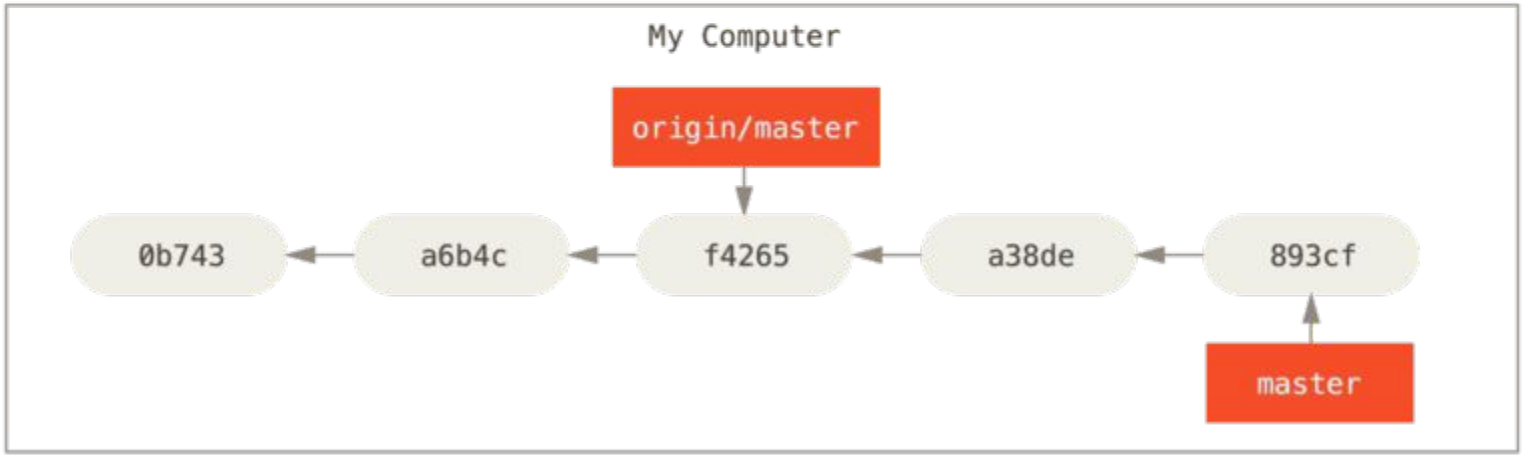
```
* remote origin
  Fetch URL: git@github.com:netology-code/devops-homeworks.git
  Push  URL: git@github.com:netology-code/devops-homeworks.git
  HEAD branch: master
  Remote branches:
    master       tracked
    version0.0 tracked
  Local branches configured for 'git pull':
    master       merges with remote master
    version0.0 merges with remote version0.0
  Local refs configured for 'git push':
    master       pushes to master     (up to date)
    version0.0  pushes to version0.0 (up to date)
```

Серверный и локальный репозиторий

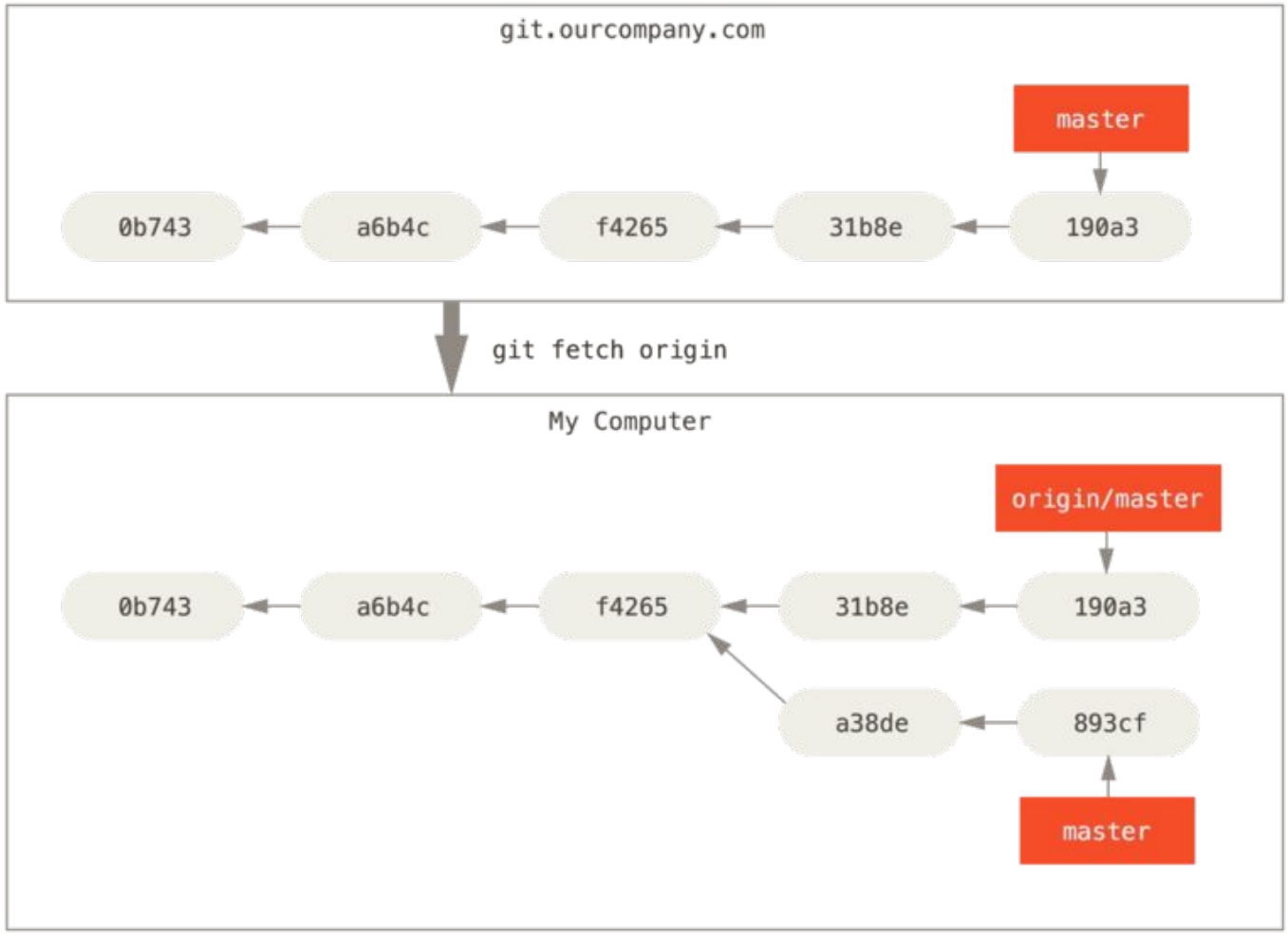


Локальная и удалённая работа могут расходиться

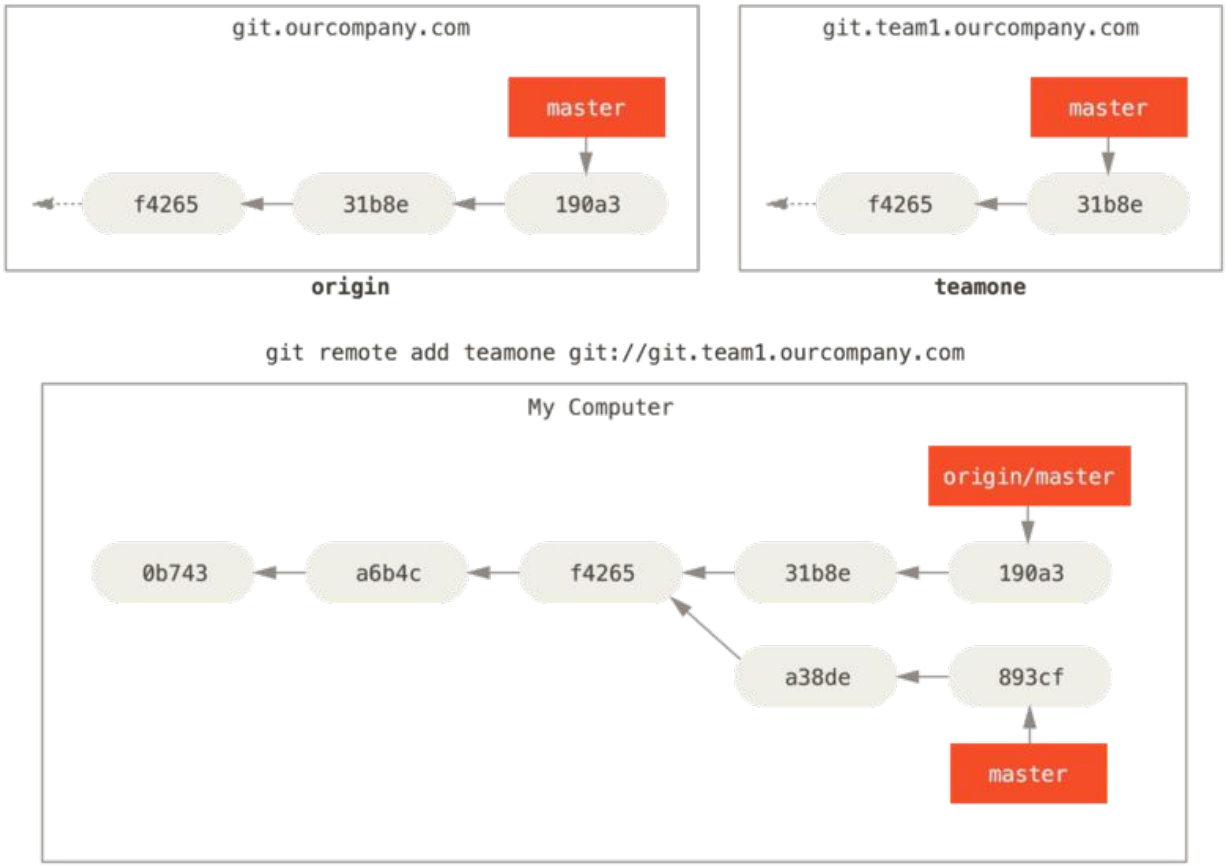




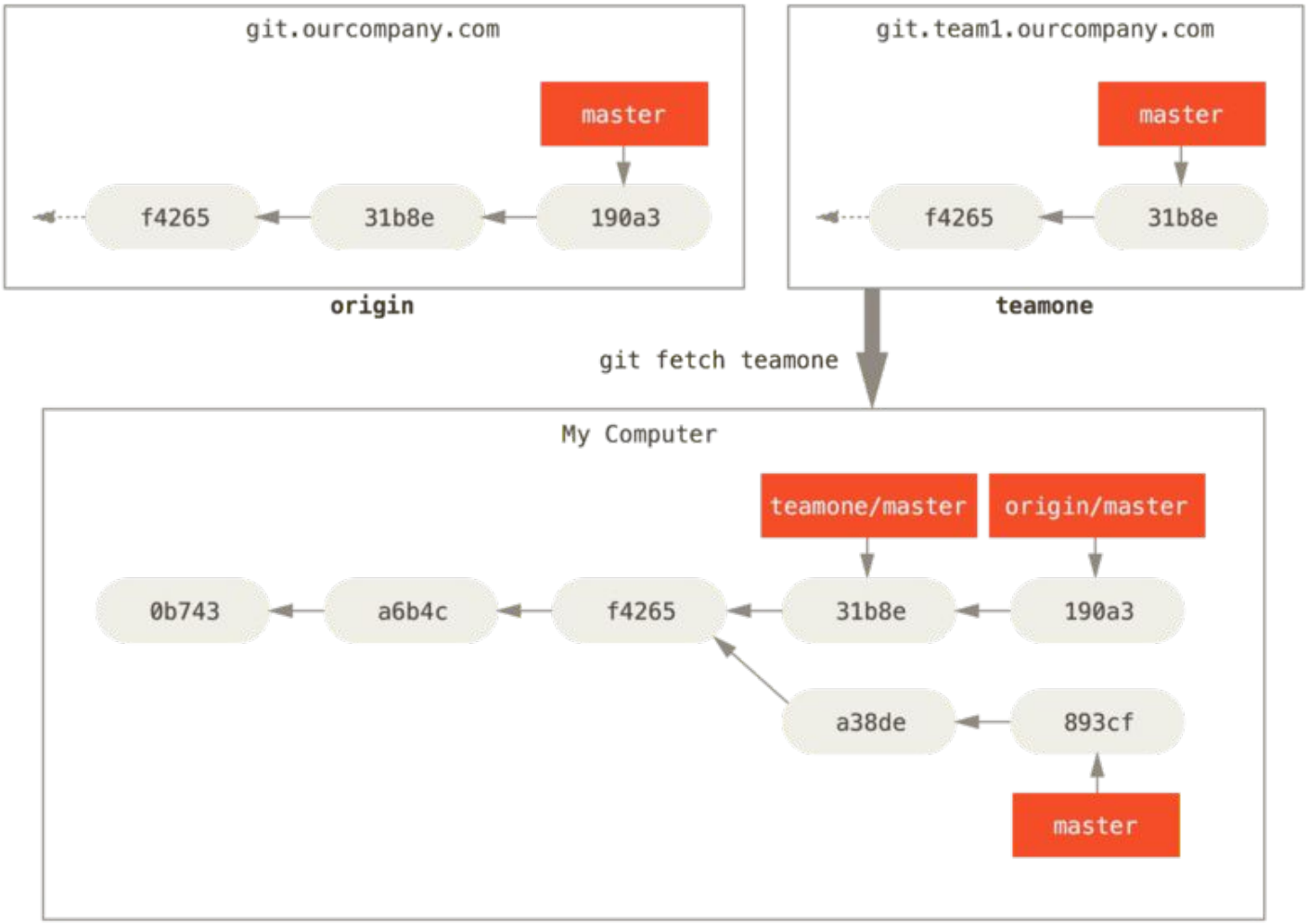
Обновление ветки слежения



Добавляем еще один удаленный репозиторий



Получаем данные из нового репозитория



Отправка изменений

$ git push <remote> <branch>

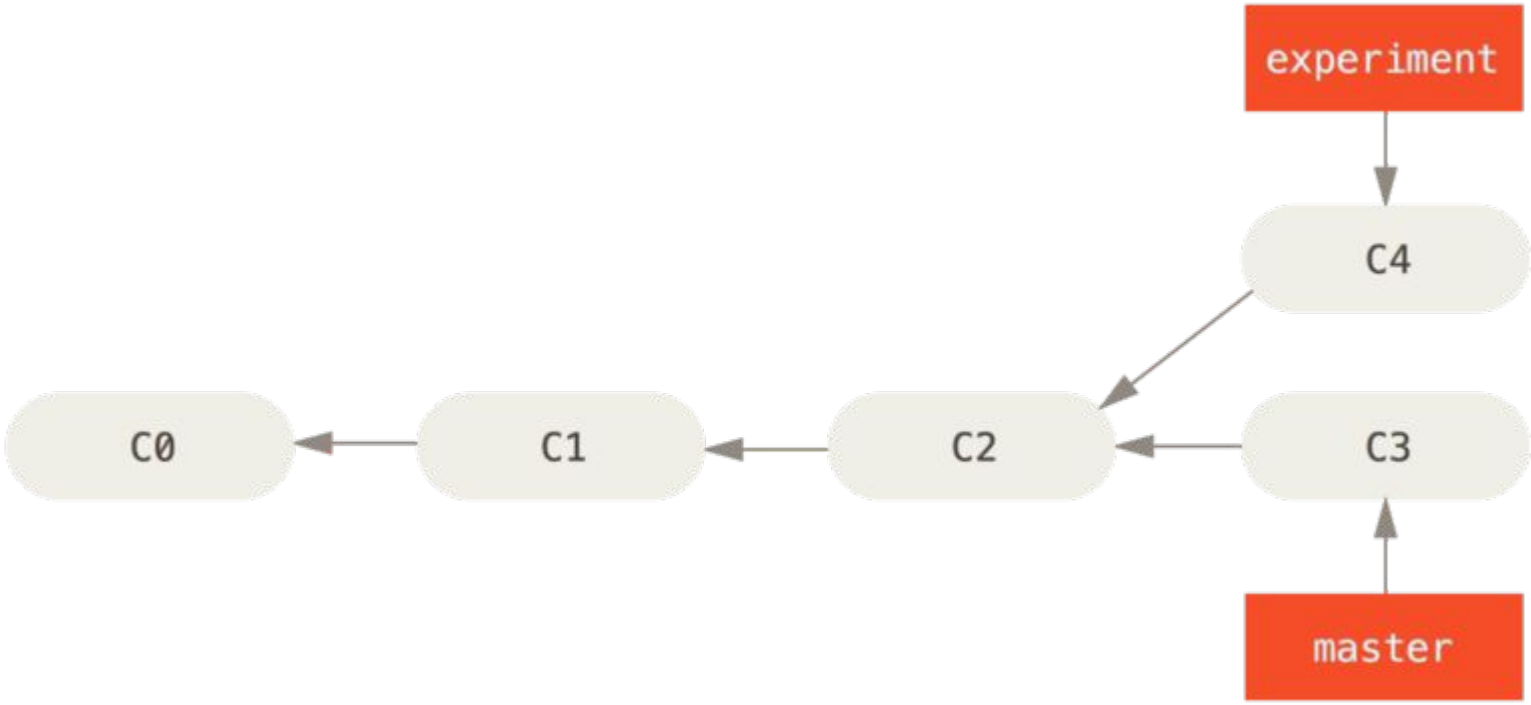
$ git push origin serverfix

```
Counting objects: 24, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (15/15), done.
Writing objects: 100% (24/24), 1.91 KiB | 0 bytes/s, done.
Total 24 (delta 2), reused 0 (delta 0)
To https://github.com/schacon/simplegit
  * [new branch]      serverfix -> serverfix
```

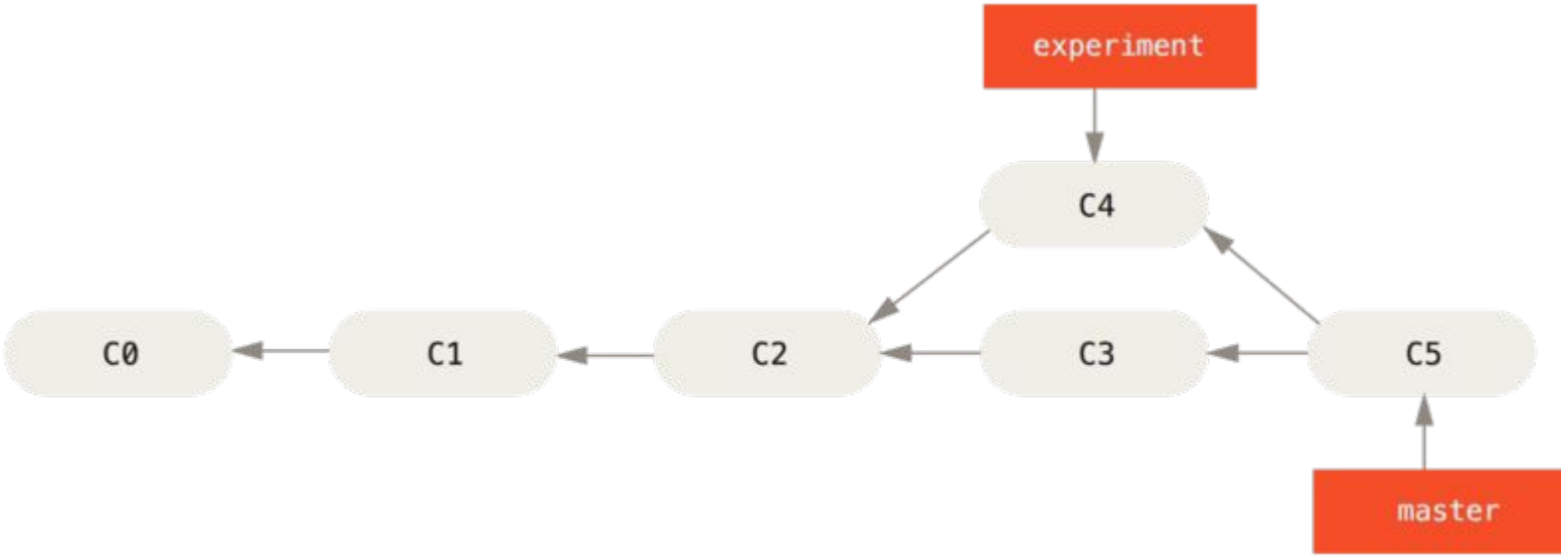
$ git push origin serverfix:awesomebranch

Rebase (перебазирование)

Исходная ситуация



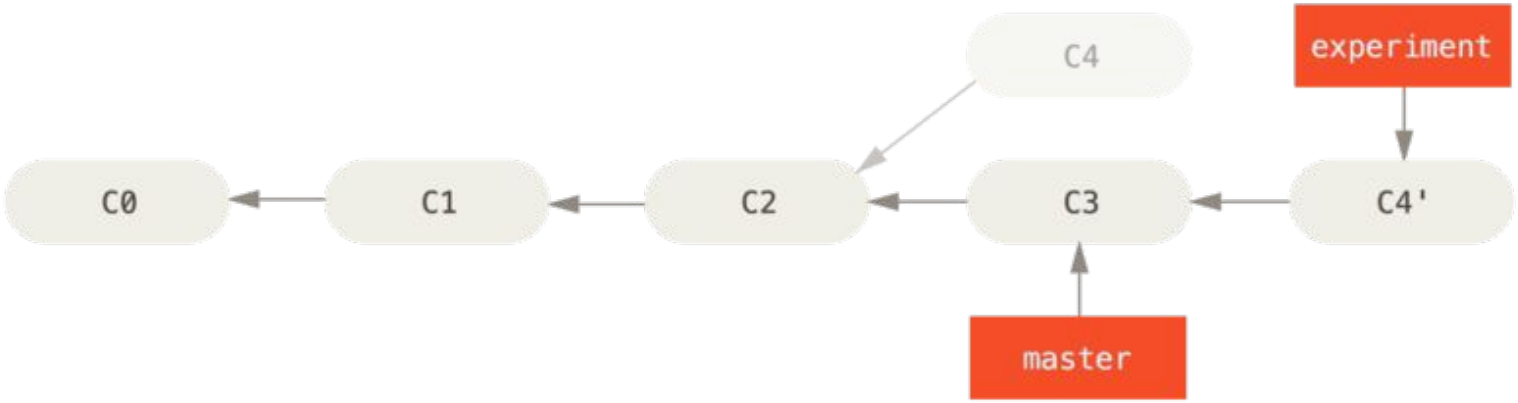
Как бы выглядел merge в мастер



Другой способ: rebase

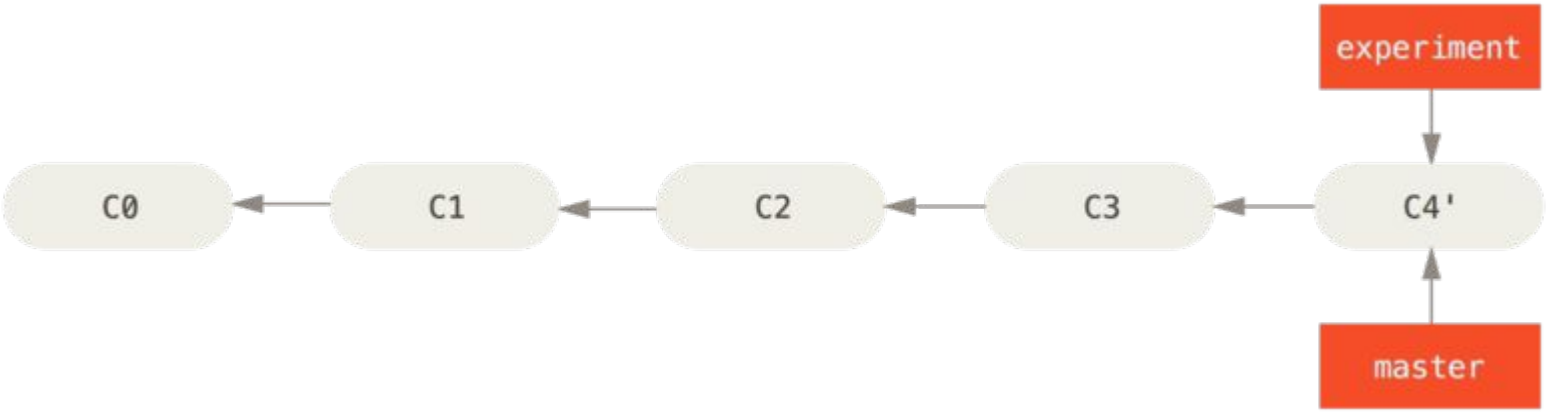
$ git checkout experiment
$ git rebase master

First, rewinding head to replay your work on top of it...
Applying: added staged command



И теперь можно делать простой merge

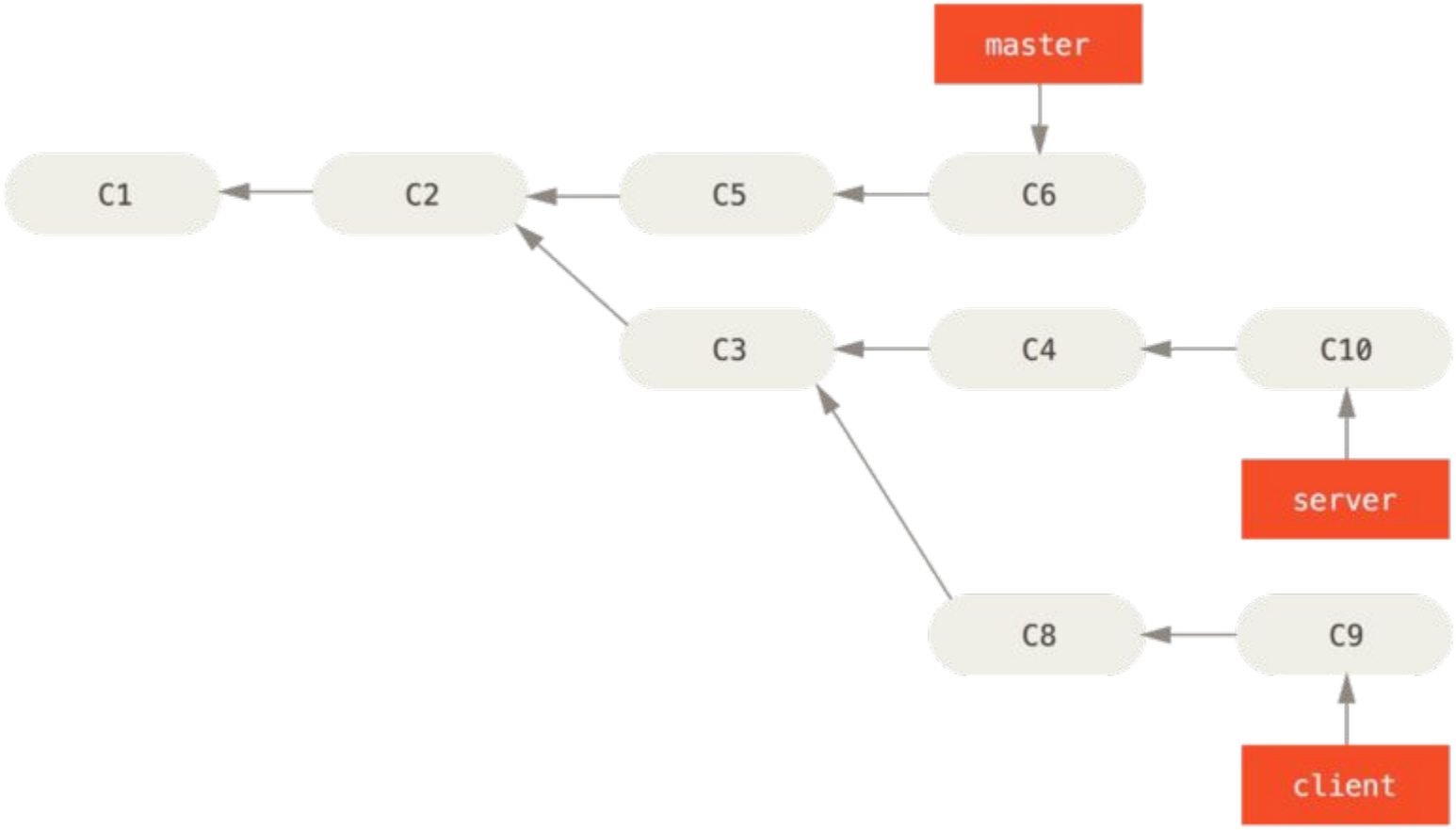
```
$ git checkout master
$ git merge experiment
```



В чем смысл rebase?

- Нет merge коммитов
- Более чистая линейная история
- Во время ребейза можно сделать еще разные интересные преобразования

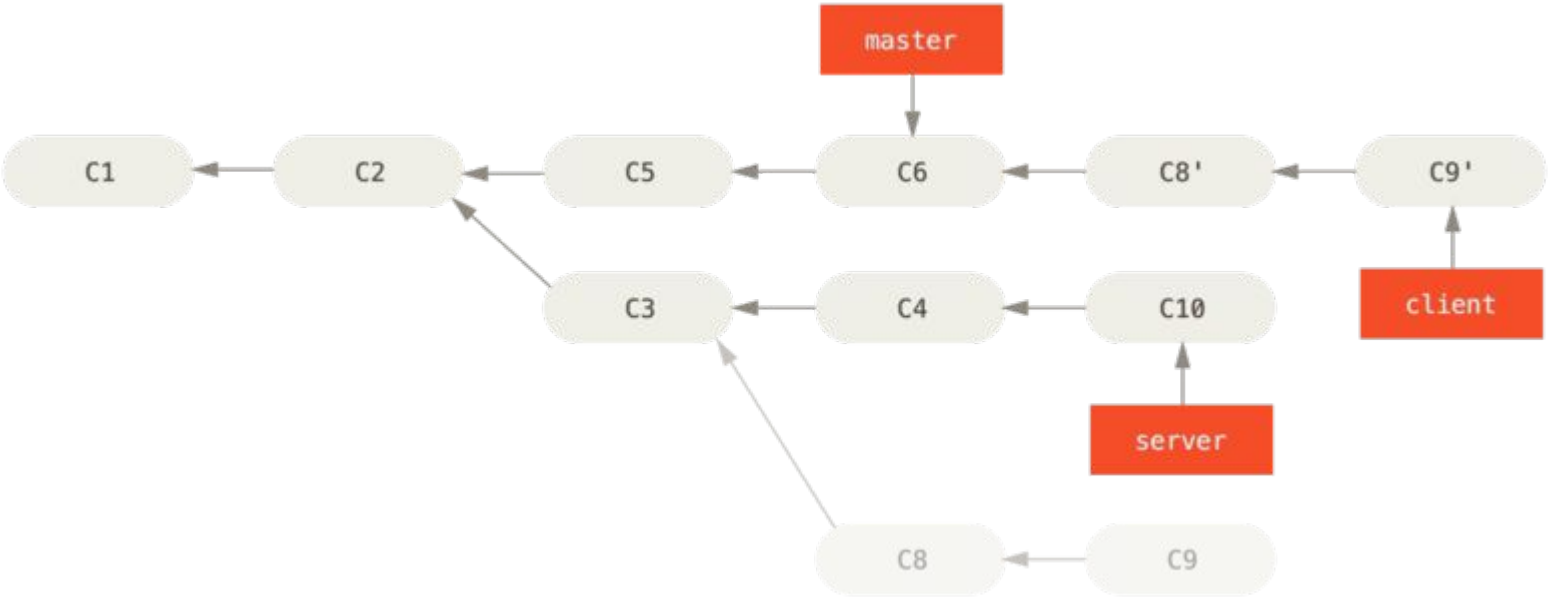
Интересный ребейз



Применяем только клиентские изменения

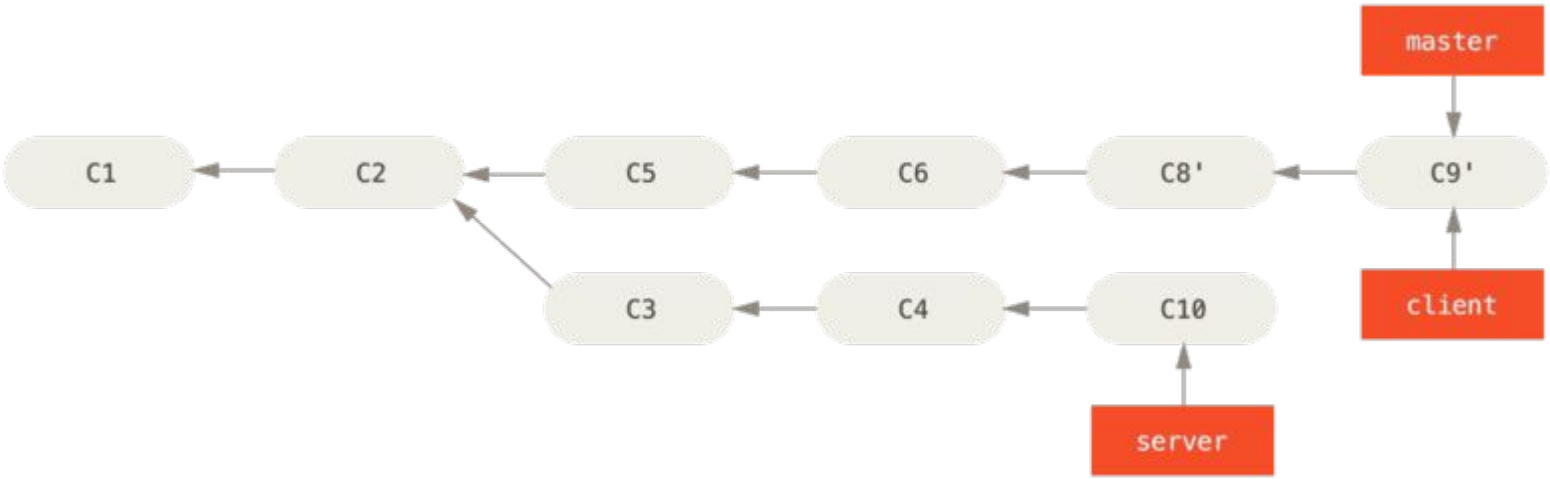
```
$ git rebase --onto master server client
```

Переключись на ветку client, найди изменения относительно ветки server и примени их для ветки master



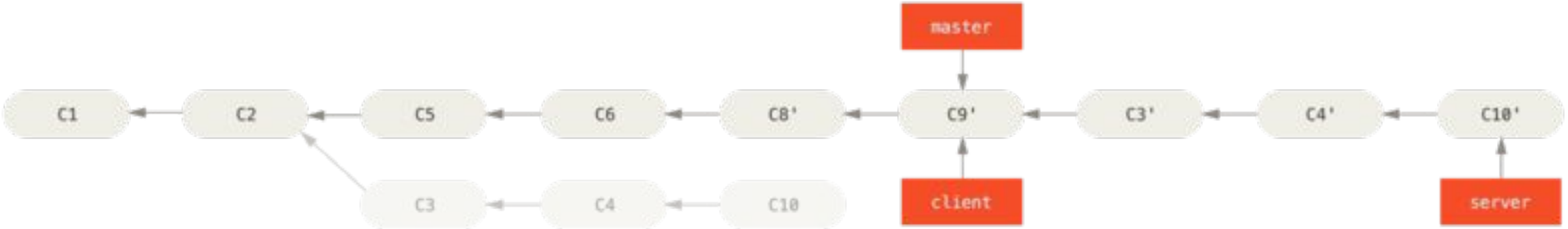
Теперь просто fast-forward мастера

```
$ git checkout master
$ git merge client
```



Ребейзим ветку server

```
$ git rebase <basebranch> <topicbranch>
$ git rebase master server
```



Окончательна история коммитов

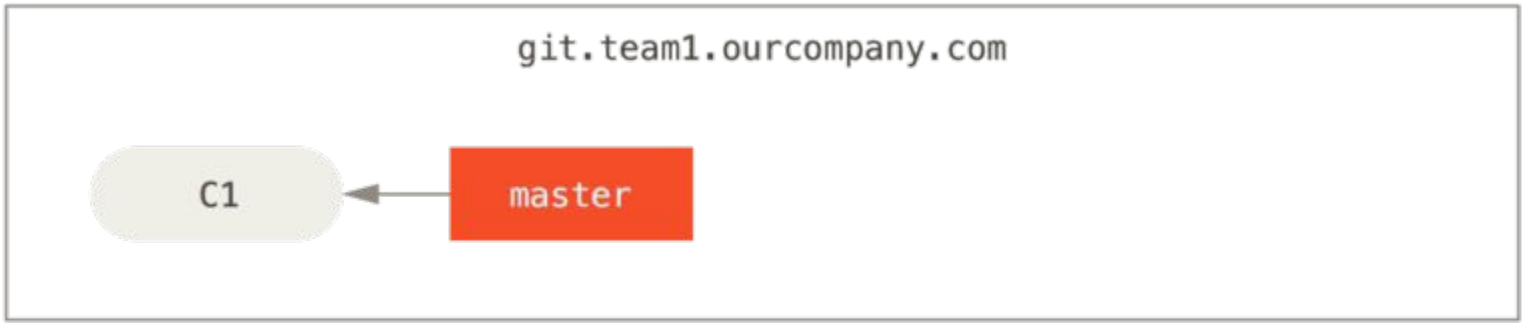
```
$ git checkout master
$ git merge server

$ git branch -d client
$ git branch -d server
```

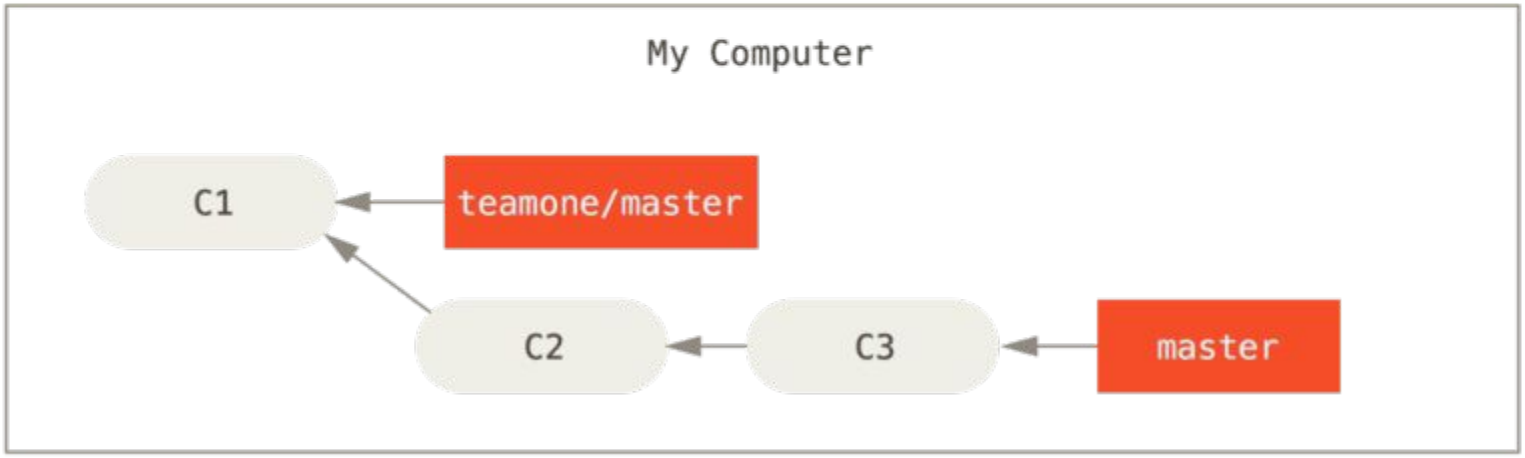
Опасности rebase

Не перемещайте коммиты уже отправленные в публичный репозиторий в общую ветку.

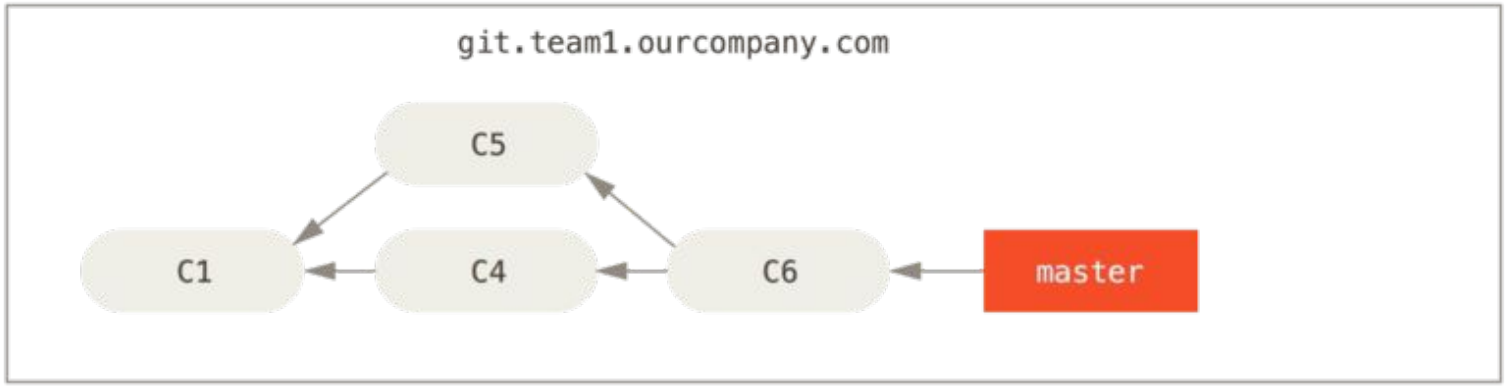
Иначе вас возненавидят.

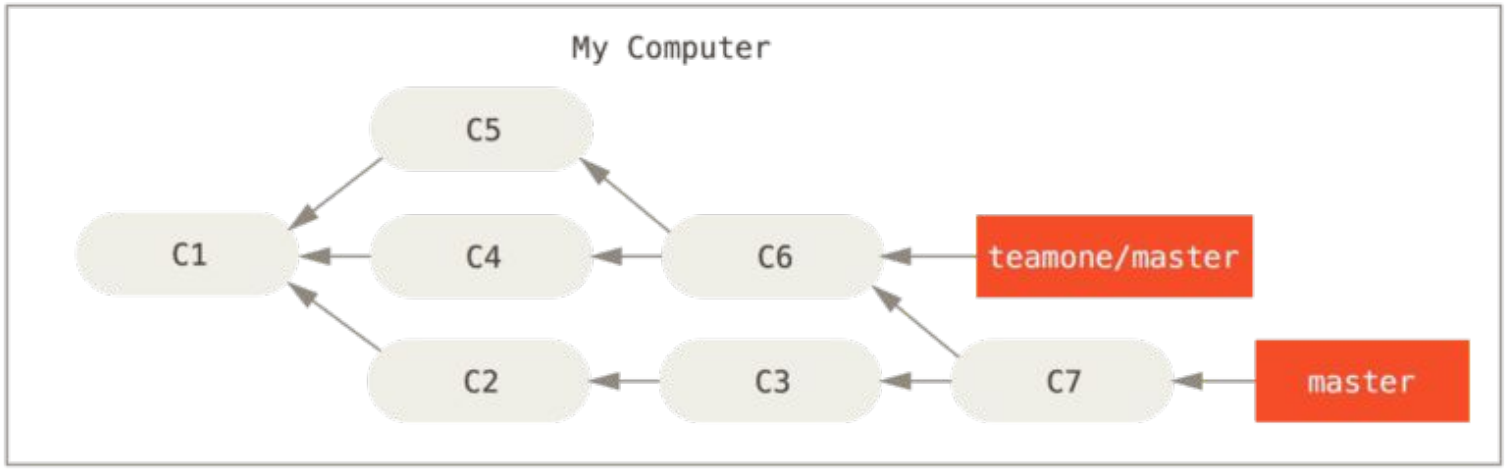
Пример для ненависти



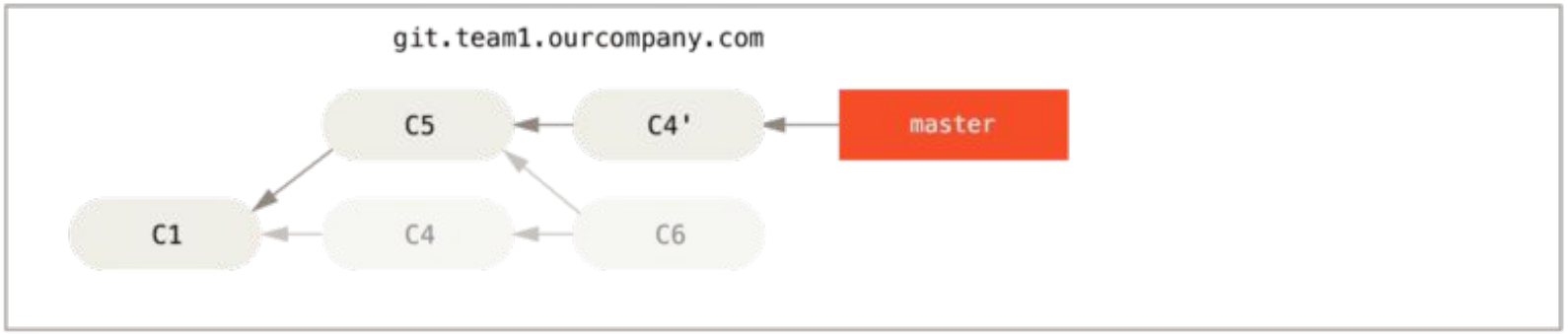


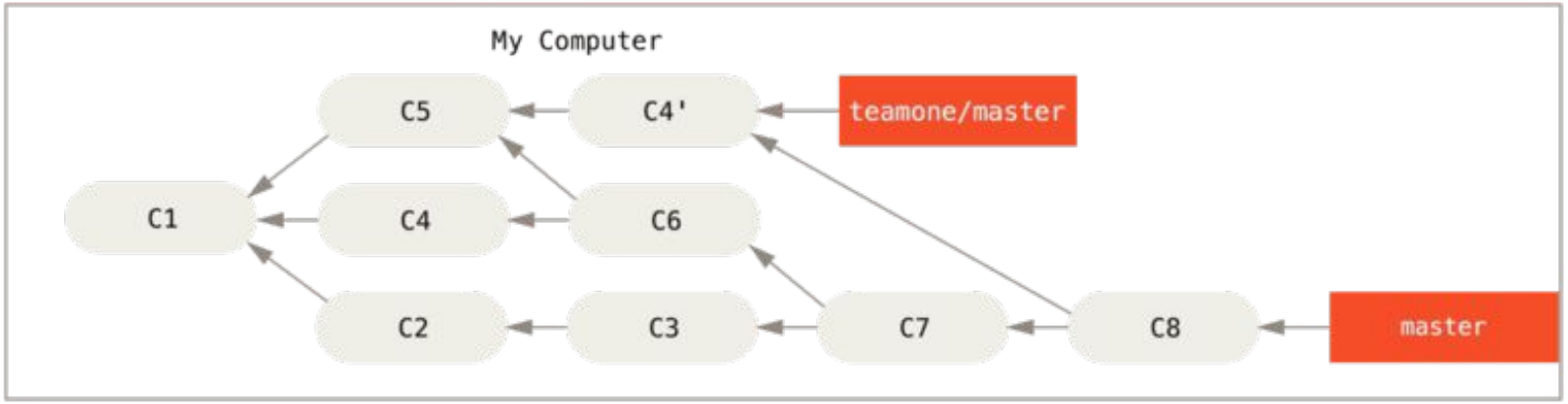
Извлекаем еще коммиты и сливаем со своей работой





Изменения коммитов на которых основывает ваша ветка





merge vs rebase

- Rebase изменяет историю
- Более чистая линейная история
- Rebase ветки которую ещё никто другой не использовал может быть неплохо – почистим мусор

Хуки

Установка хука

$ ls .git/hooks/

- applypatch-msg.sample
- fsmonitor-watchman.sample
- pre-applypatch.sample
- pre-push.sample
- pre-receive.sample
- update.sample
- commit-msg.sample
- post-update.sample

Клиентские хуки уровня коммита

Очередность запуска хуков:

1. pre-commit
2. pre-commit-msg
3. commit-msg
4. post-commit

`git commit --no-verify`, чтобы пропустить `pre-commit` и `commit-msg`

Прочие клиентские хуки

Очередность запуска хуков:

1. pre-rebase
2. post-rewrite
3. post-checkout
4. post-merge
5. pre-push

Серверные хуки

Очередность запуска хуков:

1. pre-receive
2. update
3. post-receive

Итоги

Чему мы научились

- Как работают ветки и мержи
- Работе с удаленными ветками
- Освоили инструмент Rebase
- Как использовать хуки

Домашнее задание

Домашнее задание

Давайте посмотрим ваше домашнее задание.

- Вопросы по домашней работе задавайте **в чате** мессенджера Slack.
- Задачи можно сдавать **по частям**.
- Зачёт по домашней работе проставляется после того, как **приняты все задачи**.

нетология

Задавайте вопросы и пишите отзыв о лекции!

Андрей Борю

 andrey.borue |  andrey.borue |  andreyborue