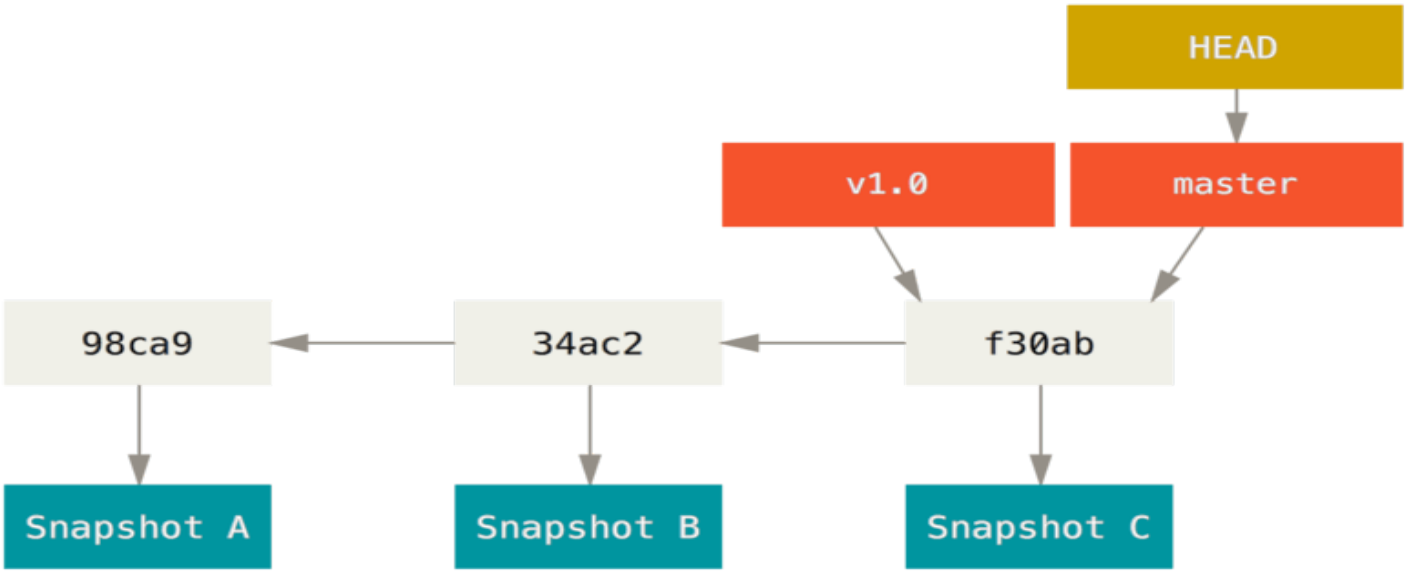
Система контроля версий Git

Ветвление продолжение

Ветка и история коммитов



Ветка в Git — это простой перемещаемый указатель на один из таких коммитов. По умолчанию, имя основной ветки в Git — *master*. Как только вы начнёте создавать коммиты, ветка *master* будет всегда указывать на последний коммит. Каждый раз при создании коммита указатель ветки *master* будет передвигаться на следующий коммит автоматически.

Операции создания и переключения

- Создать новую ветку:
  - $ git branch new_branch
- Переключится на другую ветку
  - $ git checkout other_branch
- Создать ветку и сразу переключится:
  - $ git checkout -b new_branch2

- Список имеющихся веток:

```
$ git branch
    iss53
    * master
    testing
```

- Посмотреть ветки с последними коммитами:

```
$ git branch -v
    iss53       93b412c Fix javascript issue
    * master  7a98805 Merge branch 'iss53'
    testing     782fd34 Add scott to the author list in the readme
```

Переименование ветки

Предположим, у вас есть ветка с именем *bad-branch-name*, и вы хотите изменить её на *corrected-branch-name*, сохранив при этом всю историю. Вместе с этим, вы также хотите изменить имя ветки на удалённом сервере (например GitHub). Как это сделать?

- $ git branch --move bad-branch-name corrected-branch-name

Ветка *bad-branch-name* будет переименована в *corrected-branch-name*, но это изменение пока только локальное. Чтобы все остальные увидели исправленную ветку в удалённом репозитории, отправьте её туда:

- $ git push --set-upstream origin corrected-branch-name

Однако, старая ветка всё ещё по-прежнему там, но её можно удалить с помощью команды:

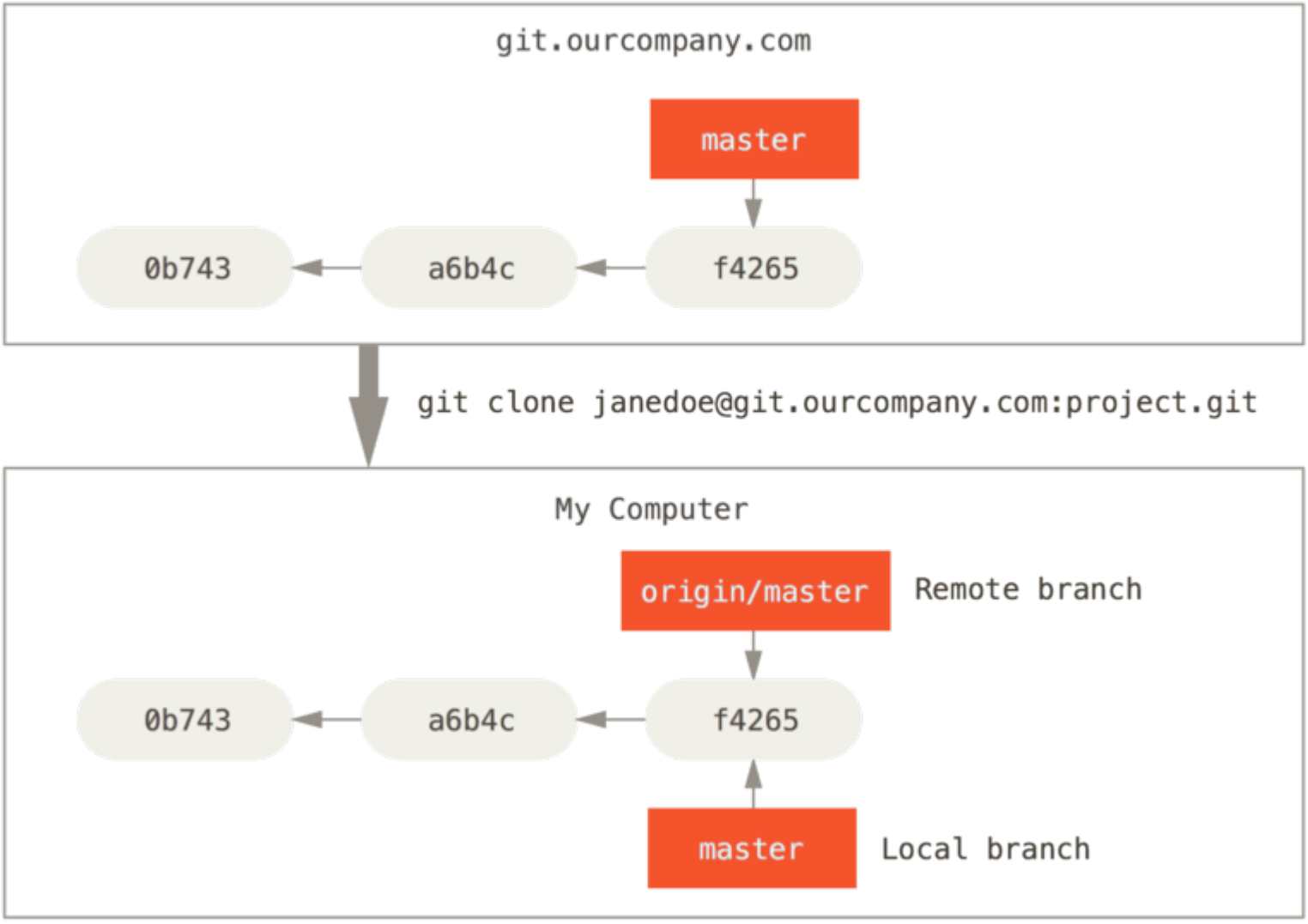
- $ git push origin --delete bad-branch-name

Удалённые ветки

- Удалённые ссылки — это ссылки (указатели) в ваших удалённых репозиториях, включая ветки, теги и так далее. Полный список удалённых ссылок можно получить с помощью команды git ls-remote <remote> или команды git remote show <remote> для получения удалённых веток и дополнительной информации. Тем не менее, более распространенным способом является использование веток слежения.
- Ветки слежения — это ссылки на определённое состояние удалённых веток. Это локальные ветки, которые нельзя перемещать; Git перемещает их автоматически при любой коммуникации с удаленным репозиторием, чтобы гарантировать точное соответствие с ним. Представляйте их как закладки для напоминания о том, где ветки в удалённых репозиториях находились во время последнего подключения к ним.
- Имена веток слежения имеют вид <remote>/<branch>. Например, если вы хотите посмотреть, как выглядела ветка *master* на сервере *origin* во время последнего соединения с ним, используйте ветку *origin/master*. Если вы с коллегой работали над одной задачей и он отправил на сервер ветку *iss53*, при том что у вас может быть своя локальная ветка *iss53*, удалённая ветка будет представлена веткой слежения с именем *origin/iss53*.

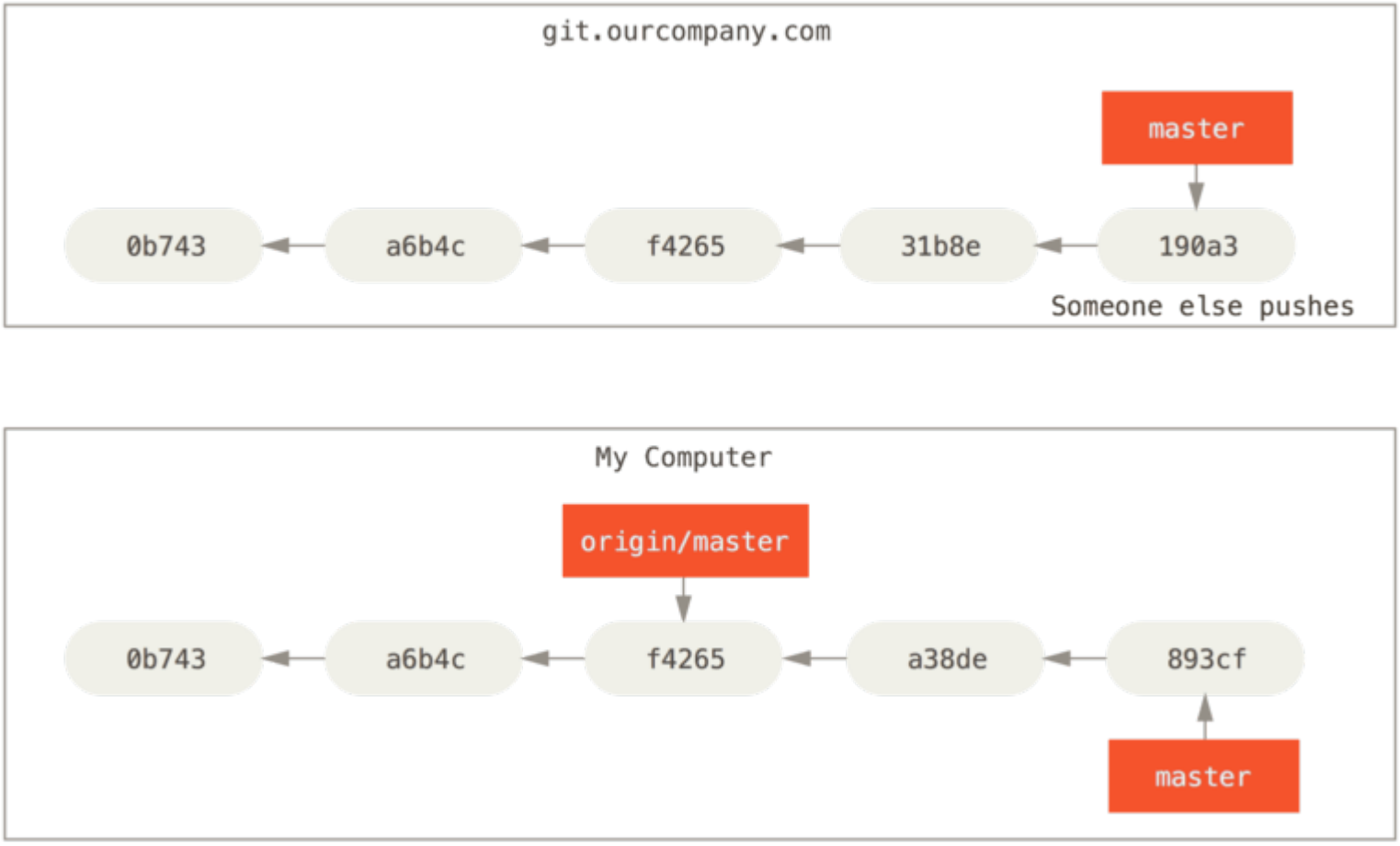
Пример

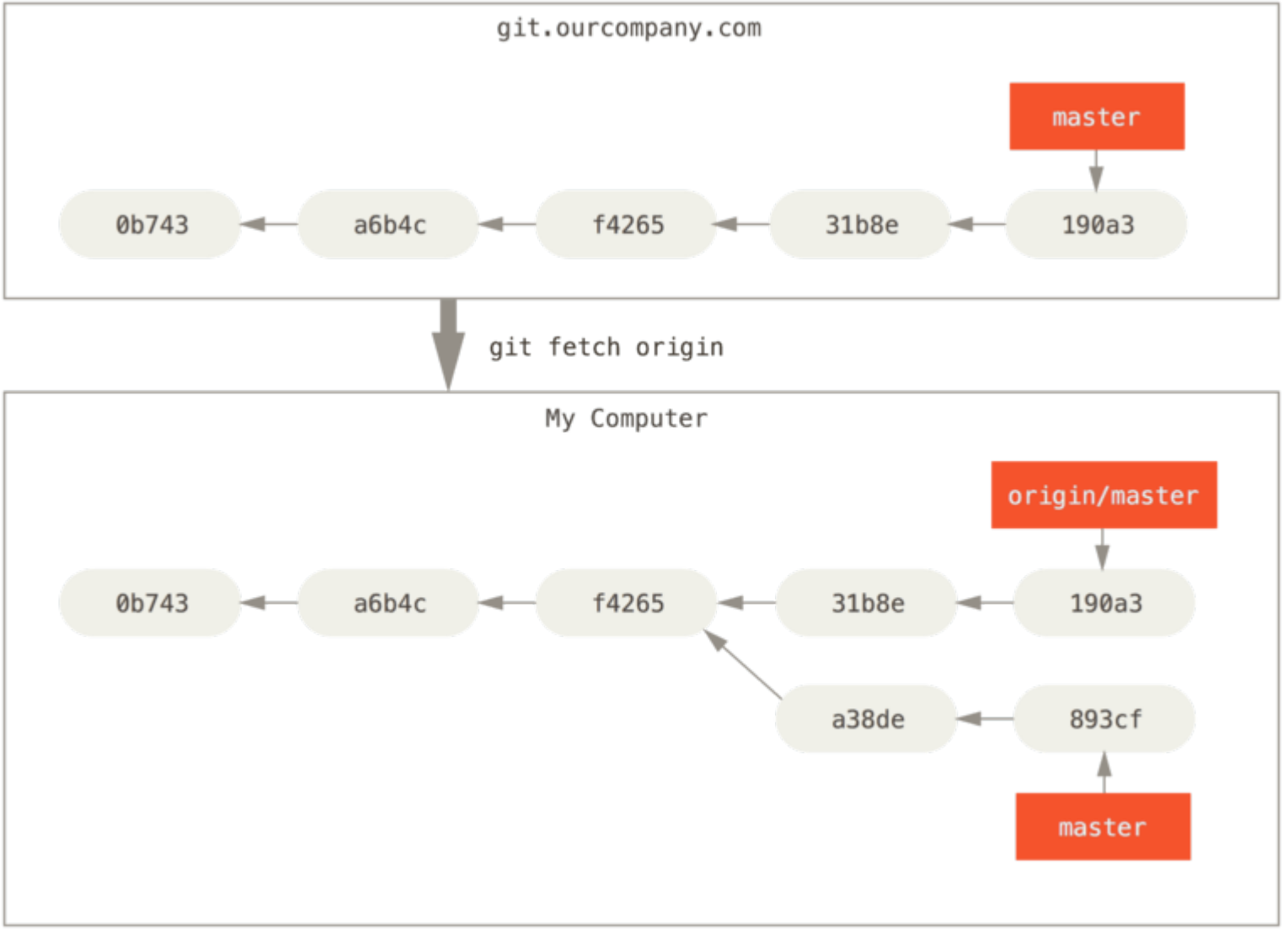
- Возможно, всё это сбивает с толку, поэтому давайте рассмотрим на примере. Скажем, у вас в сети есть свой Git-сервер с адресом *git.ourcompany.com*.
- Если вы с него что-то клонируете, команда clone автоматически назовёт его *origin*, заберёт оттуда все данные, создаст указатель на то, на что там указывает ветка *master*, и назовёт его локально *origin/master*.
- Git также создаст вам локальную ветку *master*, которая будет начинаться там же, где и ветка *master* в *origin*, так что вам будет с чего начать.


git.ourcompany.com
master
0b743
a6b4c
f4265
git clone janedoe@git.ourcompany.com:project.git
My Computer
origin/master
Remote branch
0b743
a6b4c
f4265
master
Local branch

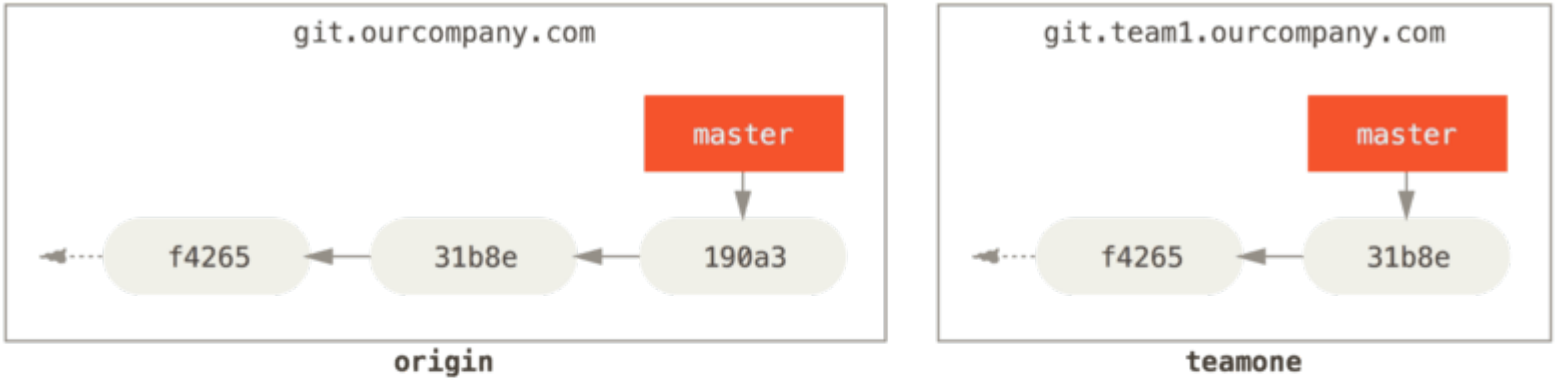
- Если вы сделаете что-то в своей локальной ветке *master*, а тем временем кто-то отправит изменения на сервер *git.ourcompany.com* и обновит там ветку master, то ваши истории продолжатся по-разному. Пока вы не свяжетесь с сервером *origin* ваш указатель *origin/master* останется на месте.

- Для синхронизации ваших изменений с удаленным сервером выполните команду git fetch <remote> (в нашем случае git fetch origin). Эта команда определяет какому серверу соответствует *«origin»* (в нашем случае это *git.ourcompany.com*), извлекает оттуда данные, которых у вас ещё нет, и обновляет локальную базу данных, сдвигая указатель *origin/master* на новую позицию.

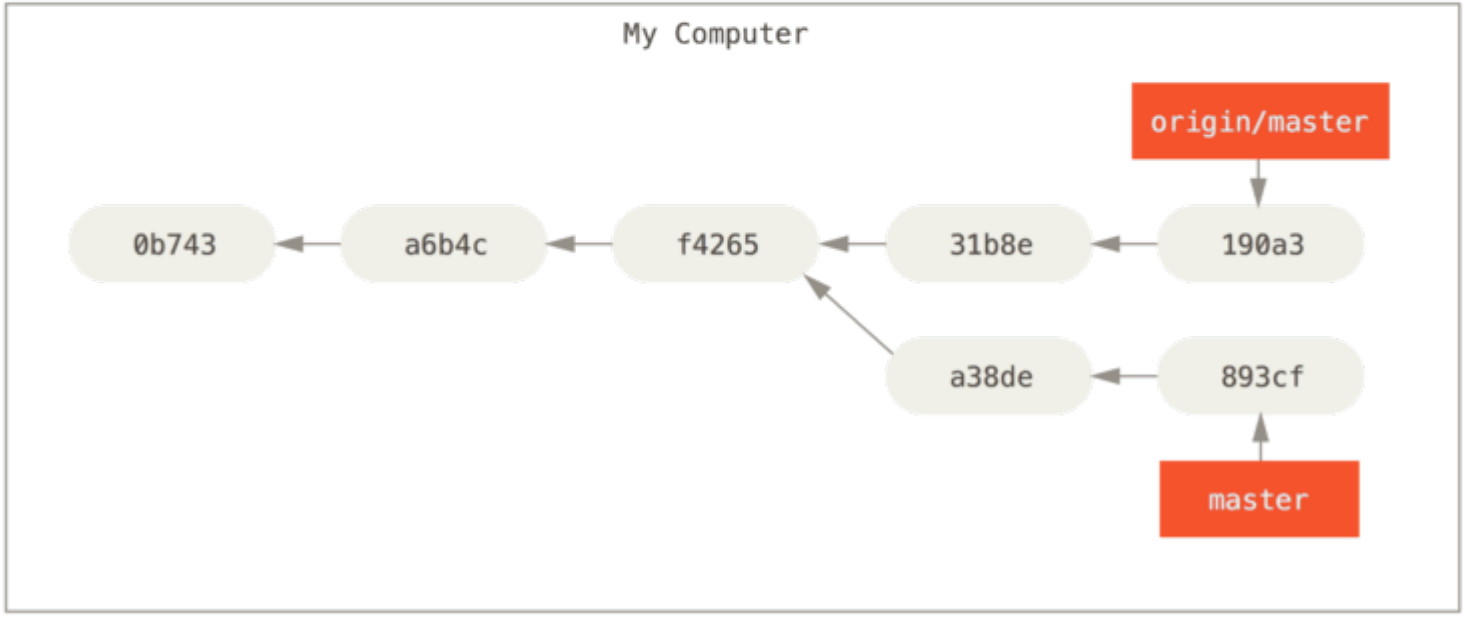

git.ourcompany.com
master
0b743
a6b4c
f4265
31b8e
190a3
Someone else pushes
My Computer
origin/master
0b743
a6b4c
f4265
a38de
893cf
master

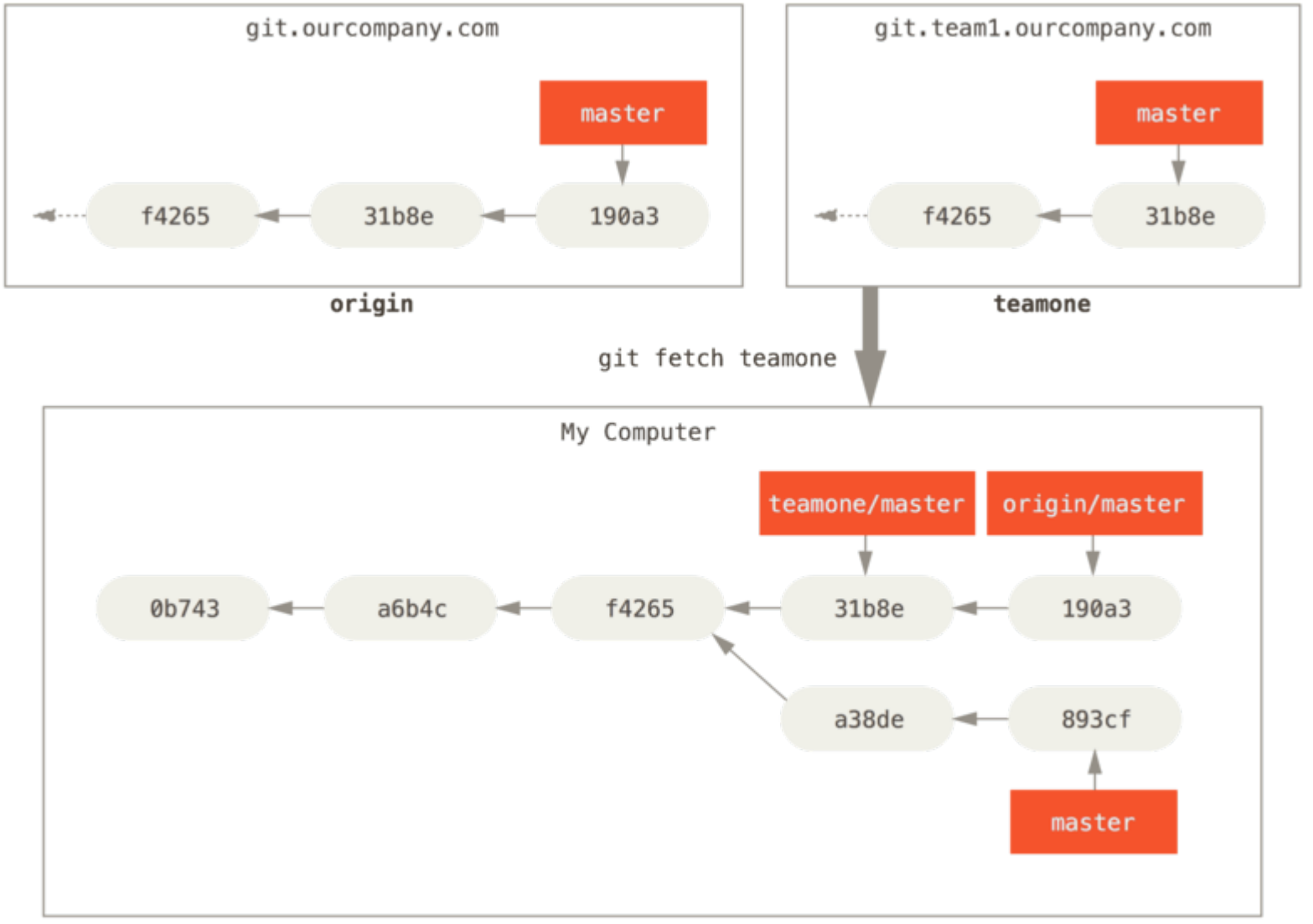

git.ourcompany.com
master
0b743
a6b4c
f4265
31b8e
190a3
git fetch origin
My Computer
origin/master
0b743
a6b4c
f4265
31b8e
190a3
a38de
893cf
master

- Чтобы продемонстрировать, как будут выглядеть удалённые ветки в ситуации с несколькими удалёнными серверами, предположим, что у вас есть ещё один внутренний Git-сервер, который используется для разработки только одной из ваших команд разработчиков. Этот сервер находится на *git.team1.ourcompany.com*. Вы можете добавить его в качестве новой удалённой ссылки для текущего проекта с помощью команды git remote add, как было описано в Основы Git. Назовите этот удалённый сервер *teamone* — это имя будет сокращением вместо полного URL.
- Теперь вы можете выполнить команду git fetch teamone для получения всех изменений с сервера *teamone*, которых у вас нет локально. Так как в данный момент на этом сервере есть только те данные, что содержит сервер *origin*, Git ничего не получит, но создаст ветку слежения с именем *teamone/master*, которая будет указывать на тот же коммит, что и ветка *master* на сервере *teamone*.



git remote add teamone git://git.team1.ourcompany.com




git.ourcompany.com
master
f4265
31b8e
190a3
origin
git.team1.ourcompany.com
master
f4265
31b8e
teamone
git fetch teamone
My Computer
teamone/master
origin/master
0b743
a6b4c
f4265
31b8e
190a3
a38de
893cf
master

Отправка изменений

- Когда вы хотите поделиться веткой, вам необходимо отправить её на удалённый сервер, где у вас есть права на запись. Ваши локальные ветки автоматически не синхронизируются с удалёнными при отправке — вам нужно явно указать те ветки, которые вы хотите отправить. Таким образом, вы можете использовать свои личные ветки для работы, которую не хотите показывать, а отправлять только те тематические ветки, над которыми вы хотите работать с кем-то совместно.

Отправка изменений

Если у вас есть ветка *serverfix*, над которой вы хотите работать с кем-то ещё, вы можете отправить её точно так же, как вы отправляли вашу первую ветку. Выполните команду 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
```

Отправка изменений

Это в некотором роде сокращение. Git автоматически разворачивает имя ветки *serverfix* до *refs/heads/serverfix:refs/heads/serverfix*, что означает «возьми мою локальную ветку serverfix и обнови ей удалённую ветку serverfix». Вы также можете выполнить

```
$ git push origin serverfix:serverfix
```

произойдёт то же самое — здесь говорится «возьми мою ветку *serverfix* и сделай её удалённой веткой *serverfix*».

Можно использовать этот формат для отправки локальной ветки в удалённую ветку с другим именем. Если вы не хотите, чтобы на удалённом сервере ветка называлась *serverfix*, то вместо предыдущей команды выполните

```
$ git push origin serverfix:awesomebranch
```

которая отправит локальную ветку *serverfix* в ветку *awesomebranch* удалённого репозитория.

Отправка изменений

- В следующий раз, когда один из ваших соавторов будет получать обновления с сервера, он получит ссылку на то, на что указывает *serverfix* на сервере, как удалённую ветку *origin/serverfix*. Необходимо отметить, что при получении данных создаются ветки слежения, вы не получаете автоматически для них локальных редактируемых копий. Другими словами, в нашем случае вы не получите новую ветку *serverfix* — только указатель *origin/serverfix*, который вы не можете изменять.
- Чтобы слить эти наработки в свою текущую рабочую ветку, выполните git merge origin/serverfix. Если вам нужна локальная ветка *serverfix*, в которой вы сможете работать, то вы можете создать её на основе ветки слежения:

```
$ git checkout -b serverfix origin/serverfix
Branch serverfix set up to track remote branch serverfix from origin.
Switched to a new branch 'serverfix'
```

Это даст вам локальную ветку, в которой можно работать и которая будет начинаться там же, где и *origin/serverfix*.

Отслеживание веток

- При клонировании репозитория, как правило, автоматически создаётся ветка *master*, которая следит за *origin/master*. Однако, при желании вы можете настроить отслеживание и других веток — следить за ветками на других серверах или отключить слежение за веткой *master*. Вы только что видели простейший пример, что сделать это можно с помощью команды git checkout -b <branch> <remote>/<branch>. Это часто используемая команда, поэтому Git предоставляет сокращённую форму записи в виде флага --track:

  $ git checkout --track origin/serverfix

  Branch serverfix set up to track remote branch serverfix from origin.

  Switched to a new branch 'serverfix'

- В действительности, это настолько распространённая команда, что существует сокращение для этого сокращения. Если вы пытаетесь извлечь ветку, которая не существует, но существует только одна удалённая ветка с точно таким же именем, то Git автоматически создаст ветку слежения:

  $ git checkout serverfix

  Branch serverfix set up to track remote branch serverfix from origin.

  Switched to a new branch 'serverfix'

Отслеживание веток

- Если вы хотите посмотреть как у вас настроены ветки слежения, воспользуйтесь опцией -vv для команды git branch. Это выведет список локальных веток и дополнительную информацию о том, какая из веток отслеживается, отстаёт, опережает или всё сразу относительно отслеживаемой.

```
$ git branch -vv
  iss53     7e424c3 [origin/iss53: ahead 2] forgot the brackets
  master    1ae2a45 [origin/master] deploying index fix
* serverfix f8674d9 [teamone/server-fix-good: ahead 3, behind 1] this should do it
  testing   5ea463a trying something new
```

- Итак, здесь мы видим, что наша ветка *iss53* следит за *origin/iss53* и «опережает» её на два изменения — это значит, что у нас есть два локальных коммита, которые не отправлены на сервер. Мы также видим, что наша ветка *master* отслеживает ветку *origin/master* и находится в актуальном состоянии. Далее мы можем видеть, что локальная ветка *serverfix* следит за веткой *server-fix-good* на сервере *teamone*, опережает её на три коммита и отстает на один — это значит, что на сервере есть один коммит, который мы ещё не слили, и три локальных коммита, которые ещё не отправлены на сервер. В конце мы видим, что наша ветка *testing* не отслеживает удаленную ветку.

Получение изменений

- Команда git fetch получает с сервера все изменения, которых у вас ещё нет, но не будет изменять состояние вашей рабочей копии. Эта команда просто получает данные и позволяет вам самостоятельно сделать слияние. Тем не менее, существует команда git pull, которая в большинстве случаев является командой git fetch, за которой непосредственно следует команда git merge. Если у вас настроена ветка слежения как показано в предыдущем разделе, или она явно установлена, или она была создана автоматически командами clone или checkout, git pull определит сервер и ветку, за которыми следит ваша текущая ветка, получит данные с этого сервера и затем попытается слить удалённую ветку.
- Обычно, лучше явно использовать команды fetch и merge, поскольку магия git pull может часто сбивать с толку.

Удаление веток

- Скажем, вы и ваши соавторы закончили с нововведением и слили его в ветку master на удалённом сервере (или в какую-то другую ветку, где хранится стабильный код). Вы можете удалить ветку на удалённом сервере используя параметр --delete для команды git push. Для удаления ветки *serverfix* на сервере, выполните следующую команду:

```
$ git push origin --delete serverfix

To https://github.com/schacon/simplegit

 - [deleted]         serverfix
```

- Всё, что делает эта строка — удаляет указатель на сервере. Как правило, Git сервер хранит данные пока не запустится сборщик мусора, поэтому если ветка была удалена случайно, чаще всего её легко восстановить.