GeekBrains

Git. Базовый курс

Урок 10

Слияние и переносы

Merge-request

Git rebase

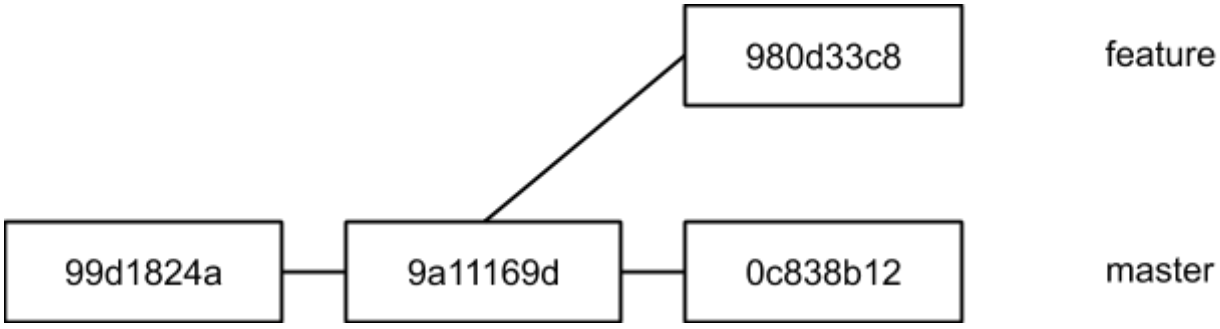
Перенос коммита

Объединение коммитов

Используемые источники

Merge-request

Давайте смоделируем ситуацию, когда мы отпочковались от ветки master и создали ветку feature, а в ней — ряд коммитов. При этом ветка master ушла вперед. Теперь нам необходимо добавить в ветку feature новые изменения.



Отпочковываем новую ветку feature:

```
$ git checkout -b feature
Switched to a new branch 'feature'
```

Давайте сразу закинем ее на сервер:

```
$ git push origin feature
```

Теперь давайте создадим в ней коммит. Вносим изменения в проект, в файле readme.txt добавляем строку 'Hello, git!'.

```
$ echo 'Hello, git!' > readme.txt
```

Добавляем изменения в виде нового коммита:

```
$ git add .
$ git commit -m 'Adding new line feature'
```

Переключаемся на ветку master:

```
$ git checkout master
```

Вносим изменение в файл hello.txt. Тоже добавляем строку 'Hello, git!'.

```
$ echo 'Hello, git!' > hello.txt
```

Создаем коммит, на этот раз в ветке master:

```
$ git add .
$ git commit -m 'Adding new line master'
```

Теперь наша задача — добавить изменения из ветки master в ветку feature. Самое простое, что можно сделать в данной ситуации — смержить ветку master в feature. Для этого переключаемся в ветку feature:

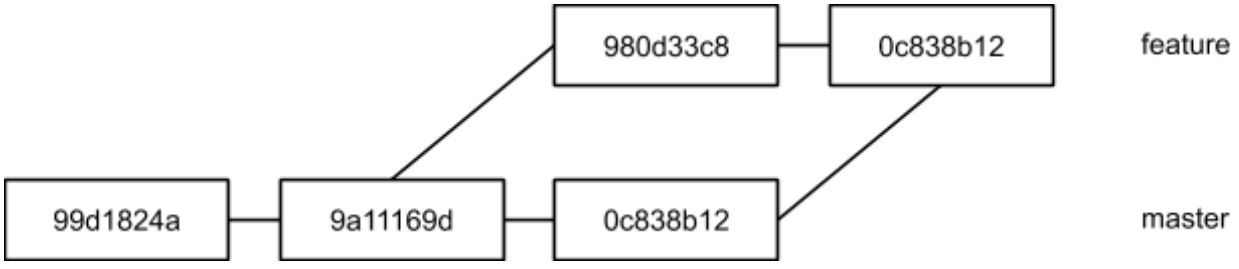
```
$ git checkout feature
```

Выполняем команду **git merge master**:

```
$ git merge master
Merge made by the 'recursive' strategy.
 hello.txt | 3 +--
 1 file changed, 1 insertion(+), 2 deletions(-)

$ git gg
*   820f627 - (HEAD -> feature) Merge branch 'master' into feature (Igor Simdyanov)
08.09.2019 23:01
|\
| * 8fe8c96 - (master) Adding new line master (Igor Simdyanov) 08.09.2019 23:00
* | 56ed0c9 - Adding new line feature (Igor Simdyanov) 08.09.2019 22:59
|/
* 089a6e8 - (tag: v1.4.0, origin/master, origin/feature, origin/HEAD) New line in
hello.php (Igor Simdyanov) 05.05.2019 13:15
...
```

Итак, изменения доставлены. У нас есть коммит из ветки feature и коммит из ветки master:



Git rebase

Существует альтернативный вариант решить задачу синхронизации изменений. Вместо того, чтобы вливать в ветку feature изменения из ветки master, коммиты ветки feature перемещаются таким образом, чтобы ветка feature начиналась там, где заканчивается ветка master. В результате ветка feature содержит все коммиты master плюс новые коммиты.

Добиться этого можно при помощи операции git rebase.

Благодаря такому переносу веток все изменения в master будут доступны в ветке feature. При этом читать историю коммитов становится гораздо проще. Если над проектом работает сразу много человек, параллельных веток может быть много. Операция переноса позволяет вытянуть историю в более или менее линейную последовательность.

Операция git rebase считается сложной, к тому же, изменяющей историю git, поэтому во многих компаниях она запрещена. Тем не менее, знать о ней необходимо.

Переключаемся в ветку master:

```
$ git checkout master
```

Давайте попробуем снова воспроизвести ситуацию с веткой feature, добавив в ветку master новый коммит.

Добавляем в файл hello.txt строку 'Hello, git rebase!':

```
$ echo 'Hello, git rebase!' > hello.txt
```

Создаем коммит в ветке master:

```
$ git add .
$ git commit -m 'Adding new hello git rebase line master'
```

Для того, чтобы совершить процедуру rebase, перемещаемся в ветку feature:

```
$ git checkout feature
$ git diff master
diff --git a/hello.txt b/hello.txt
index 4399e32..6566899 100644
--- a/hello.txt
+++ b/hello.txt
@@ -1 +1 @@
-Hello, git rebase!
+Hello, git!
diff --git a/readme.txt b/readme.txt
index 91bd8b0..6566899 100644
--- a/readme.txt
+++ b/readme.txt
@@ -1,2 +1 @@
-Новый проект
-Другая строка
+Hello, git!
```

Выполняем команду **git rebase**:

```
$ git rebase master
First, rewinding head to replay your work on top of it...
Applying: Adding new line feature
```

В конце команды указываем название ветки master. Теперь ветка feature перемещена вперед.

```
$ git gg
* 7118826 - (HEAD -> feature) Adding new line feature (Igor Simdyanov) 08.09.2019
23:03
* d49d411 - (master) Adding new hello git rebase line master (Igor Simdyanov)
08.09.2019 23:02
* 8fe8c96 - Adding new line master (Igor Simdyanov) 08.09.2019 23:00
* 089a6e8 - (tag: v1.4.0, origin/master, origin/feature, origin/HEAD) New line in
hello.php (Igor Simdyanov) 05.05.2019 13:15
...
```

Если мы попробуем сейчас закинуть ветку на сервер, у нас ничего не получится:

```
$ git push origin feature
```

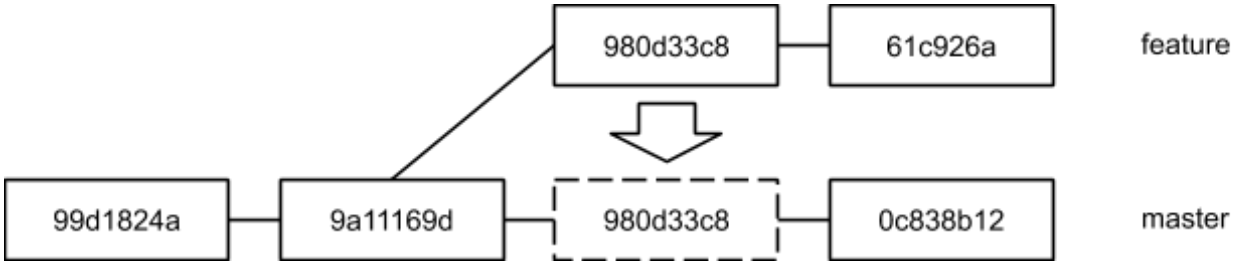
Это связано с тем, что на сервере по-прежнему хранится старый вариант ветки feature. Чтобы забросить новый вариант, после команды **rebase** потребуется принудительно перезаписать ветку:

```
$ git push origin feature -f
```

Для этого команде **git push** передается параметр -f, от английского force. Считается, что такое принудительное перезаписывание ветки потенциально опасно и при неумелом использовании может повредить проект. Каждая команда договаривается, пользоваться ли **rebase** или избегать ее.

Перенос коммита

Иногда в ветке может быть много изменений, а сливать одну в другую нельзя. Например, не настало время выкатки изменений, тем не менее, нам может потребоваться перенести один или несколько коммитов из одной ветки в другую. Для этого предназначена команда **git cherry-pick**.



Давайте создадим коммит в ветке feature, который мы потом будем переносить в ветку master. Изменим файл readme.txt, добавив в него строку 'Hello, git rebase!'

Создаем коммит:

```
$ git status
$ git add .
$ git commit -m 'Adding new line git rebase feature'
```

Давайте посмотрим его хэш:

```
$ git gg
```

Помещаем его в буфер обмена и переключаемся на ветку master:

```
$ git checkout master
```

Команда для переноса коммита всегда выполняется в ветке, в которую мы переносим коммит:

```
$ git cherry-pick ce268e51
```

Выполняем команду cherry-pick, указывая ей в качестве параметра хэш переносимого коммита.

При попытке слияния у нас возникает конфликт. Давайте поправим его. Для этого отправляемся в редактор и открываем файл readme.txt

Устраняем конфликт и отправляемся в консоль:

```
$ git status
```

Git предлагает нам выполнение команды **cherry-pick**, для этого нужно добавить параметр **--continue**. Добавляем файл в индекс:

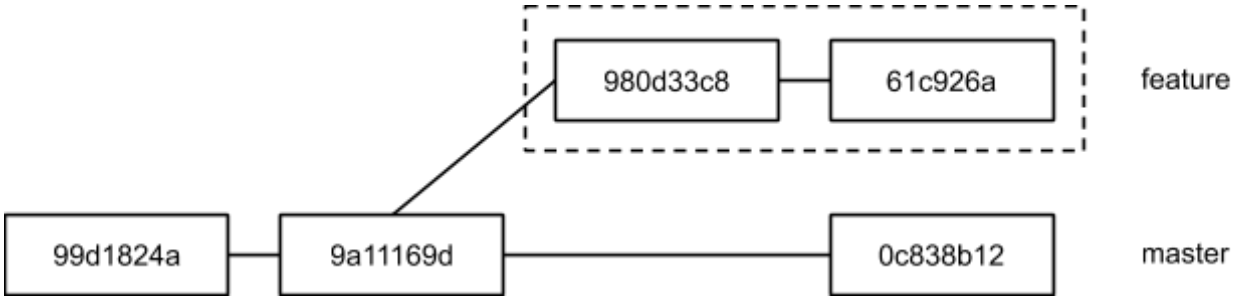
```
$ git add .
```

И завершаем перенос коммита:

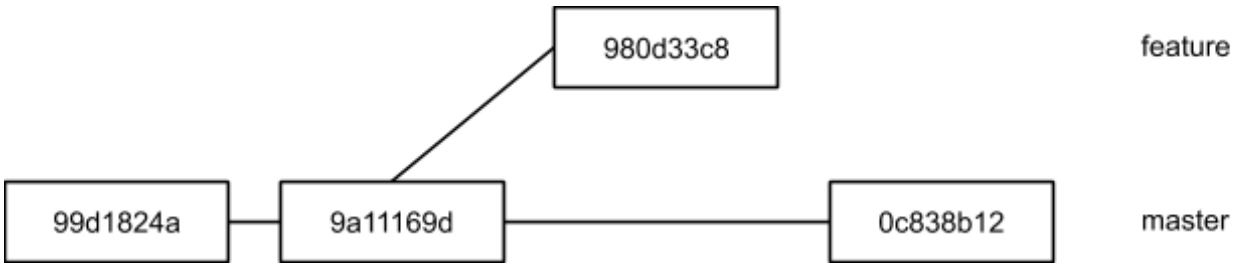
```
$ git cherry-pick --continue
```

В ветку master был перенесен коммит, однако его хэш поменялся. Несмотря на то, что вносимые изменения совпадают с изменениями исходного, это новый коммит со своим собственным хэшем.

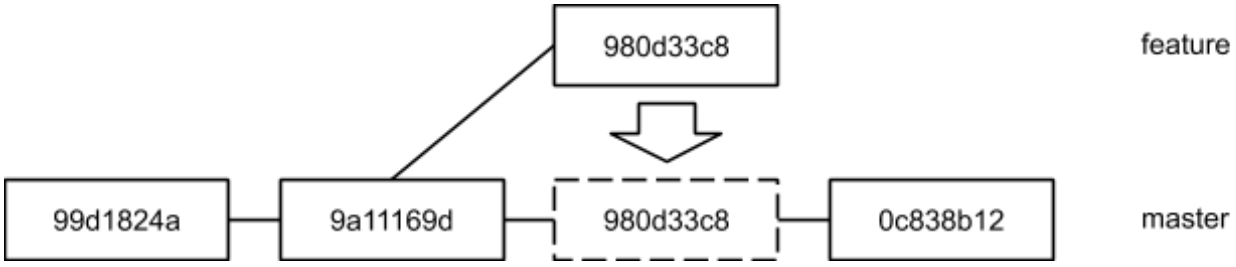
Объединение коммитов



Иногда коммитов, которые необходимо перенести из одной ветки в другую, довольно много. Переносить их поодиночке командой **cherry-pick** может быть утомительно.



В этом случае несколько коммитов ветки предварительно объединяют в один:



Который затем и переносят в другую ветку при помощи команды **cherry-pick**.

Для операции объединения нескольких коммитов в один используется команда **git rebase -i.** Давайте переключимся на ветку feature:

```
$ git checkout feature
```

Создадим еще какой-то коммит. Добавляем в файл readme.txt строку 'Hello, squash!' и создаем коммит:

```
$ git status
$ git add .
$ git commit -m 'Adding new line'
```

Теперь выполняем команду **git cherry**, чтобы выяснить, насколько ветка feature отличается от ветки master:

```
$ git cherry -v master
```

Как видно, она отличается на три коммита. Мы знаем, что сослаться на последний коммит ветки можно при помощи готового тэга HEAD:

```
$ git show HEAD
```

Если нам потребуется коммит, который предшествует последнему, можно через тильду добавить значение 1:

```
$ git show HEAD~1
```

Увеличивая цифру после тильды, можно перебрать все коммиты ветки:

```
$ git show HEAD~2
```

Давайте снова выполним команду **git cherry**:

```
$ git cherry -v master
```

Так как разница составляет три коммита, начальный коммит — HEAD~3:

```
$ git show HEAD~3
```

Для того, чтобы объединить коммиты, начиная с этого коммита до последнего, выполняем команду **git rebase -i**:

```
$ git rebase -i HEAD~3
```

Открывается файл с перечислением всех коммитов и описанием, что можно с ними сделать. Если перед каждым коммитом мы оставляем значение pick — они остаются, если squash — объединяются.

У вас должен остаться хотя бы один коммит, помеченный как pick. Сохраняем файл. Как результат, git склеивает коммиты в один. Давайте в этом убедимся:

```
$ git cherry -v master
```

У нас ветка feature отличается от master только одним коммитом. Если мы сейчас попробуем отправить ветку на сервер, у нас ничего не получится:

```
$ git push origin feature
```

Мы воспользовались перезаписыванием истории проекта при помощи команды **git rebase**, поэтому нам потребуется перезаписать ветку при с использованием параметра **-f** или **--force**:

```
$ git push origin feature --force
```

Или в сокращенном варианте — просто **-f**:

```
$ git push origin feature -f
```

Несколько коммитов объединены в один. Теперь можно переносить коммит в другую ветку при помощи команды **git cherry pick** или просто смержить ветку.

Используемые источники

Для подготовки методического пособия мы использовали эти ресурсы:

- Официальная документация Git: **https://git-scm.com/book/ru/v2.**