Devpractice
Разработка программного обеспечения, технологии и наука

Git для начинающих. Часть 9. Как удалить коммит в git?

Автор: Marat Abdrakhmanov | 01.04.2018 2 комментария

Рассмотрим довольно важный вопрос: как удалить коммит в *git*? Начнем с вопроса отмены изменений в рабочей директории, после этого перейдем к репозиторию. В рамках этой темы изучим вопросы удаления и замены последнего коммита, работу с отдельными файлами и использование команд *git revert* и *git reset*.

Отмена изменений в файлах в рабочей директории

Если вы сделали какие-то изменения в файле и хотите вернуть предыдущий вариант, то для этого следует обратиться к репозиторию и взять из него файл, с которым вы работаете. Таким образом, в вашу рабочую директорию будет скопирован файл из репозитория с заменой. Например, вы работаете с файлом *main.c* и внесли в него какие-то изменения. Для того чтобы вернуться к предыдущей версии (последней отправленной в репозиторий) воспользуйтесь командой *git checkout*.

```
> git checkout -- main.c
```

Ключ “- -” означает, что нас интересует файл в текущем бранче (ветке).

Отмена коммитов в *git*

Работа с последним коммитом

Для демонстранции возможностей *git* создадим новый каталог и инициализируем в нем репозиторий.

```
> git init
```

Добавим в каталог файл *main.c*.

```
> touch main.c
```

Отправим изменения в репозиторий.

```
> git add main.c
> git commit -m "first commit"
[master (root-commit) 86f1495] first commit
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 main.c
```

Внесем изменения в файл.

```
> echo "// main.c file" > main.c
```

И сделаем еще один коммит.

```
> git add main.c
> git commit -m "second commit"
[master d142679] second commit
 1 file changed, 1 insertion(+)
```

В репозиторий, на данный момент, было сделано два коммита.

```
> git log --oneline
d142679 second commit
86f1495 first commit
```

Теперь удалим последний коммит и вместо него отправим другой. Предварительно изменим содержимое файла *main.c*.

```
> echo "// author: Writer" > main.c
```

Отправим изменения в репозиторий с заметой последнего коммита.

```
> git add main.c
> git commit --amend -m "third commit"
```

```
> git log --oneline
18411fd third commit
86f1495 first commit
```

Как вы можете видеть: из репозитория пропал коммит с *id=d142679*, вместо него теперь коммит с *id=18411fd*.

Отмена изменений в файле в выбранном коммите

Сделаем ещё несколько изменений в нашем файле *main.c*, каждое из которых будет фиксироваться коммитом в репозиторий.

```
> echo "// Some text 1" > main.c
> git add main.c
> git commit -m "fourth commit"
[master dcf7253 ] fourth commit
 1 file changed, 1 insertion(+), 1 deletion(-)

> echo "// Some text 2" > main.c
> git add main.c
> git commit -m "fifth commit"
[master 7f2eb3a ] fifth commit
 1 file changed, 1 insertion(+), 1 deletion(-)

> git log --oneline
7f2eb3a fifth commit
dcf7253 fourth commit
18411fd third commit
86f1495 first commit
```

Помните, что в предыдущем разделе мы поменяли коммит с сообщением “second commit” на “third commit”, поэтому он идет сразу после “first commit”.

Представим ситуацию, что два последних коммита были неправильными, и нам нужно вернуться к версии *18411fd* и внести изменения именно в нее. В нашем примере, мы работаем только с одним файлом, но в реальном проекте файлов будет много, и после коммитов, в рамках которых вы внесли изменения в интересующий вас файл, может быть ещё довольно много коммитов, фиксирующих изменения в других файлах. Просто так взять и удалить коммиты из середины ветки не получится – это нарушит связность, что идет в разрез с идеологией *git*. Одни из возможных вариантов – это получить версию файла из нужного нам коммита, внести в него изменения и сделать новый коммит. Для начала посмотрим на содержимое файла *main.c* из последнего, на текущий момент, коммита.

```
> git checkout main.c
> cat main.c
// Some text 2
```

Для просмотра содержимого файла в коммите с *id=18411fd* воспользуемся правилами работы с *tree-ish* (об этом подробно написано здесь)

```
> git show 18411fd:main.c
// author: Writer
```

Переместим в рабочую директорию файл *main.c* из репозитория с коммитом *id=18411fd*.

```
> git checkout 18411fd -- main.c
> cat main.c
// author: Writer
```

Мы видим, что теперь содержимое файла *main.c* соответствует тому, что было на момент создания коммита с *id=18411fd*. Сделаем коммит в репозиторий и в сообщении укажем, что он отменяет два предыдущих.

```
> git add main.c
> git commit -m "return main.c from third commit"
[master cffc5ad] return main.c from third commit
 1 file changed, 1 insertion(+), 1 deletion(-)

> git log --oneline
cffc5ad return main.c from third commit
7f2eb3a fifth commit
dcf7253 fourth commit
18411fd third commit
86f1495 first commit
```

Таким образом мы вернулись к предыдущей версии файла *main.c* и при этом сохранили всю историю изменений.

Использование git revert для быстрой отмены изменений

Рассмотрим ещё одни способ отмены коммитов, на этот раз воспользуемся командой *git revert*.

В нашем примере, отменим коммит с *id=cffc5ad*. После того как вы введете команду *git revert* (см. ниже), система *git* выдаст сообщение в текстовом редакторе, если вы согласны с тем, что будет написано в открытом файле, то просто сохраните его и закройте. В результате изменения будут применены, и автоматически сформируется и отправится в репозиторий коммит.

```
> git revert cffc5ad
[master 81499da] Revert "return main.c from third commit"
 1 file changed, 1 insertion(+), 1 deletion(-)
```

Если вы хотите поменять редактор, то воспользуйтесь командой.

```
> git config core.editor "notepad.exe"
```

Обратите внимание, что в этом случае будут изменены настройки для текущего репозитория. Более подробно об изменении настроек смотрите в “Git для начинающих. Часть 3. Настройка Git”

Проверим, применялась ли настройка.

```
> git config core.editor
notepad.exe
```

Посмотрим на список коммитов в репозитории.

```
> git log --oneline
81499da Revert "return main.c from third commit"
cffc5ad return main.c from third commit
7f2eb3a fifth commit
dcf7253 fourth commit
18411fd third commit
86f1495 first commit
```

Содержимое файла вернулось к тому, что было сделано в рамках коммита с *id=7f2eb3a.*

```
> cat main.c
// Some text 2

> git show 7f2eb3a:main.c
// Some text 2
```

Отмена группы коммитов

ВНИМАНИЕ! Используйте эту команду очень аккуратно!

Если вы не знакомы с концепцией указателя *HEAD*, то обязательно прочитайте статью “*Git* для начинающих. Часть 7. Поговорим о *HEAD* и *tree-ish*“. *HEAD* указывает на коммит в репозитории, с которого будет вестись дальнейшая запись, т.е. на родителя следующего коммита. Существует три опции, которые можно использовать с командой *git reset* для изменения положения *HEAD* и управления состоянием *stage* и рабочей директории, сейчас мы все это подробно разберем.

Удаление коммитов из репозитория (без изменения рабочей директории) (ключ *–soft*)

Для изменения положения указателя *HEAD* в репозитории, без оказания влияния рабочую директорию (в *stage,* при этом, будет зафиксированно отличие рабочей директории от репозитория), используйте ключ *–soft*. Посмотрим ещё раз на наш репозиторий.

```
> git log --oneline
81499da Revert "return main.c from third commit"
cffc5ad return main.c from third commit
7f2eb3a fifth commit
dcf7253 fourth commit
```

```
18411fd third commit
86f1495 first commit
```

Содержимое файла *main.c* в рабочей директории.

```
> cat main.c
// Some text 2
```

Содержимое файла *main.c* в репозитории.

```
> git show HEAD:main.c
// Some text 2
```

Теперь переместим *HEAD* в репозитории на коммит с *id=dcf7253.*

```
> git reset --soft dcf7253
```

Получим следующий список коммитов.

```
> git log --oneline
dcf7253 fourth commit
18411fd third commit
86f1495 first commit
```

Содержимое файла *main.c* в репозитории выглядит так.

```
> git show HEAD:main.c
// Some text 1
```

В рабочей директории файл *main.c* остался прежним (эти изменения отправлены в *stage*).

```
> cat main.c
// Some text 2
```

Для того, чтобы зафиксировать в репозитории последнее состояние файла *main.c* сделаем коммит.

```
> git commit -m "soft reset example"
[master db1a8b0] soft reset example
 1 file changed, 1 insertion(+), 1 deletion(-)
```

Посмотрим на список коммитов.

```
> git log --oneline
db1a8b0 soft reset example
```

```
dcf7253 fourth commit
18411fd third commit
86f1495 first commit
```

Как видите из репозитория пропали следующие коммиты:

```
81499da Revert "return main.c from third commit"
cffc5ad return main.c from third commit
7f2eb3a fifth commit
```

Удаление коммитов из репозитория и очистка *stage* (без изменения рабочей директории) (ключ *–mixed*)

Если использовать команду *git reset* с аргументом *–mixed*, то в репозитории указатель *HEAD* переместится на нужный коммит, а также будет сброшено содержимое *stage*. Отменим последний коммит.

```
> git reset --mixed dcf7253
Unstaged changes after reset:
M       main.c
```

В результате изменилось содержимое репозитория.

```
> git log --oneline
dcf7253 fourth commit
18411fd third commit
86f1495 first commit
```

Содержимое файла *main.c* в последнем коммите выглядит так.

```
> git show HEAD:main.c
// Some text 1
```

Файл *main.c* в рабочей директории не изменился.

```
> cat main.c
// Some text 2
```

Отправим изменения вначале в *stage*, а потом в репозиторий.

```
> git add main.c
> git commit -m "mixed reset example"
[master ab4ef00] mixed reset example
 1 file changed, 1 insertion(+), 1 deletion(-)
```

Удаление коммитов из репозитория, очистка *stage* и внесение изменений в рабочую директорию (ключ *–hard*)

Если вы воспользуетесь ключем *–hard*, то обратного пути уже не будет. Вы не сможете восстановить данные из рабочей директории. Все компоненты *git* (репозиторий, *stage* и рабочая директория) будут приведены к одному виду в соответствии с коммитом, на который будет перенесен указатель *HEAD*.

Текущее содержимое репозитория выглядит так.

```
> git log --oneline
ab4ef00 mixed reset example
dcf7253 fourth commit
18411fd third commit
86f1495 first commit
```

Посмотрим на содержимое файла *main.c* в каталоге и репозитории.

```
> cat main.c
// Some text 2
> git show HEAD:main.c
// Some text 2
```

Содержимое файлов идентично.

Удалим все коммиты до самого первого с *id=86f1495.*

```
> git reset --hard 86f1495
HEAD is now at 86f1495 first commit
```

Содержимое репозитория.

```
> git log --oneline
86f1495 first commit
```

Состояние рабочей директории и *stage*.

```
> git status
On branch master
nothing to commit, working tree clean
```

Содержимое файла *main.c* в репозитории и в рабочей директории.

```
> cat main.c
> git show HEAD:main.c
```

Файл *main.c* пуст.

Т.к. мы воспользовались командой *git reset* с ключем *–hard*, то восстановить прежнее состояние нам не получится.

БУДЬТЕ ОЧЕНЬ АККУРАТНЫ С ЭТОЙ КОМАНДОЙ!

> *Отличный курс по git делают ребята из **GeekBrains**, найдите в разделе “Курсы” курс “Git. Быстрый старт”, он **бесплатный**!*

<<< Часть 8. Добавление, удаление и переименование файлов в репозитории

Раздел: Git Git для начинающих Метки: Git, Git для начинающих, Уроки по Git

Git для начинающих. Часть 9. Как удалить коммит в git?: 2 комментария

Paul
12.04.2018

Работа с последним коммитом:
git commit –amend -m “third commit”

Это отменяет последний коммит, но не возвращает файл в исходное состояние?..
А в какой задаче это может быть полезно?

adminka
14.04.2018

Эта команда перезаписывает последний коммит (т.е. последний удаляется и на его место встает новый), файл в исходное состояние не возвращается.

> *А в какой задаче это может быть полезно?*

хм… хороший вопрос! Даже затрудняюсь на него ответить. Ну например, вы закоммитили какой-то “ужас” и не хотите, чтобы он стал достоянием общественности))