Как вернуться (откатиться) к более раннему коммиту?

Вопрос задан 7 лет назад Изменён 4 года 8 месяцев назад Просмотрен 546k раз

154

133

Я хочу вернуться к более раннему коммиту. Как мне это сделать?

Вот что показывает команда `git log`:

```
$ git log
commit dddddd
Author: Me <me@me.com>
Date:   Thu Nov 4 18:59:41 2010 -0400

Add buzz

commit cccccc
Author: Me <me@me.com>
Date:   Thu Nov 4 05:13:39 2010 -0400

Add fizz

commit bbbbbb
Author: Me <me@me.com>
Date:   Thu Nov 4 00:55:06 2010 -0400

Add bar

commit aaaaaa
Author: Me <me@me.com>
Date:   Wed Nov 3 23:56:08 2010 -0400

Add foo
```

git | git-commit | git-checkout | git-reset | git-revert

Поделиться Улучшить вопрос Отслеживать

изменён 23 июн 2015 в 9:50

ixSci
23.5k ● 3 ● 42 ● 59

задан 23 июн 2015 в 9:43

Nick Volynkin ♦
32.6k ● 24 ● 124 ● 211

См. также: Убрать последний git-коммит, не меняя файлов проекта – Nick Volynkin ♦ 14 июн 2016 в 19:40

ассоциация: stackoverflow.com/questions/927358/... – Nicolas Chabanovsky ♦ 2 мар 2017 в 12:59

1 ответ

Сортировка: Наивысший рейтинг (по умолчанию)

332

Этот вопрос можно понять по-разному:

1. Что значит вернуться или откатиться: просто посмотреть, изменить содержимое рабочей области, изменить историю Git?
2. Что именно откатить: рабочую область (worktree), индекс (область подготовки коммита, staging area), текущую ветку, удаленную ветку?

3. К какой позиции откатить: к индексу, к последнему коммиту, к произвольному коммиту?

Обозначим начальную ситуацию на следующей схеме:

```
              (i) (wt)
A - B - C - D - ? - ?
            ↑
          master
          (HEAD)
```

`A`, `B`, `C`, `D` — коммиты в ветке `master`.
`(HEAD)` — местоположение указателя HEAD.
`(i)` — состояние индекса Git. Если совпадает с `(HEAD)` - пуст. Если нет - содержит изменения, подготовленные к следующему коммиту.
`(wt)` — состояние рабочей области проекта (working tree). Если совпадает с `(i)` — нет неиндексированных изменений, если не совпадает — есть изменения.
`↑` обозначает коммит, на который указывает определенная ветка или указатель.

Вот решения, в зависимости от задачи:

1. Временно переключиться на другой коммит

Если вам нужно просто переключиться на другой коммит, чтобы, например, посмотреть на его содержимое, достаточно команды `git checkout`:

```
git checkout aaaaaa

 (wt)
 (i)
  A - B - C - D
  ↑           ↑
(HEAD)    master
```

Сейчас репозиторий находится в состоянии «detached HEAD». Чтобы переключиться обратно, используйте имя ветки (например, `master`):

```
git checkout master
```

2. Переключиться на коммит и продолжить работу с него

Если вы хотите продолжить работу с другого коммита, вам понадобится новая ветка. Можно переключиться и создать ее одной командой:

```
git checkout -b имя-новой-ветки aaaaaa

 (wt)
 (i)
  A - B - C - D
  ↑           ↑
 new        master
(HEAD)
```

3. Удалить изменения в рабочей области и вернуть ее к состоянию как при последнем коммите.

Начальное состояние:

```
              (i) (wt)
A - B - C - D - ? - ?
            ↑
          master
          (HEAD)
```

3.1 Безопасно — с помощью кармана (stash)

3.1.1 Только неиндексированные

Можно ~~удалить~~ прикарманить только те изменения, которые еще не были индексированы (командой `add`):

```
git stash save --keep-index
```

Конечное состояние:

```
              (wt)
              (i)
A - B - C - D - ?         ?
            ↑             ↑
          master      stash{0}
          (HEAD)
```

3.1.2 Индексированные и нет

Эта команда отменяет все индексированные и неиндексированные изменения в рабочей области, сохраняя их в карман (stash).

```
git stash save
```

Конечное состояние:

```
           (wt)
           (i)
A - B - C - D             ?
            ↑             ↑
          master      stash{0}
          (HEAD)
```

> **Восстановление несохраненных изменений**: легко и просто.

```
git stash apply
```

Если stash совсем не нужен, его можно удалить.

```
# удалить последнюю запись кармана
git stash drop
```

Подробнее про использование stash.

После этого восстановить изменения всё ещё можно, но сложнее: How to recover a dropped stash in Git?

3.2 Опасный способ

> **Осторожно!** Эта команда **безвозвратно удаляет несохраненные текущие изменения из рабочей области и из индекса** Если они вам все-таки нужны, воспользуйтесь `git stash`.
>
> **Восстановление несохраненных изменений**: неиндексированные потеряны полностью, но вы можете восстановить то, что было проиндексировано.

Здесь мы будем использовать `git reset --hard`

Выполняем:

```
git reset --hard HEAD
```

Конечное состояние:

```
           (wt)
           (i)
A - B - C - D - x - x
            ↑
          master
          (HEAD)
```

4. Перейти к более раннему коммиту в текущей ветке и удалить из нее все последующие (неопубликованные)

> **Осторожно!** Эта команда переписывает историю Git-репозитория. **Если вы уже опубликовали (`git push`) свои изменения, то этот способ использовать нельзя (см. почему)**. Используйте вариант из пункта 5 (`git revert`).

4.1 При этом сохранить изменения в индекс репозитория:

```
git reset --soft bbbbbb
```

После этого индекс репозитория будет содержать все изменения от `cccccc` до `dddddd`. Теперь вы можете сделать новый коммит (или несколько) на основе этих изменений.

```
          (wt)
          (i)
A - B - C - D
    ↑
  master
  (HEAD)
```

4.2 Сохранить изменения в рабочей области, но не в индексе.

```
git reset bbbbbb
```

Эта команда просто перемещает указатель ветки, но не отражает изменения в индексе (он будет пустым).

```
   (i)    (wt)
A - B - C - D
    ↑
  master
  (HEAD)
```

4.3 Просто выбросить изменения.

> **Осторожно!** Эта команда **безвозвратно удаляет несохраненные текущие изменения**. Если удаляемые коммиты не принадлежат никакой другой ветке, то они тоже будут потеряны.
>
> **Восстановление коммитов**: Используйте `git reflog` и этот вопрос чтобы найти и восстановить коммиты; иначе сборщик мусора удалит их безвозвратно через некоторое время.
>
> **Восстановление несохраненных изменений**: неиндексированные потеряны полностью, но вы можете восстановить то, что было проиндексировано.

Начальное состояние:

```
              (i) (wt)
A - B - C - D - ? -  ?
          ↑
        master
        (HEAD)
```

Выполняем:

```
git reset --hard bbbbbb
```

Конечное состояние:

```
   (wt)
   (i)
A - B - C - D - x - x
```

```
    ↑
  master
  (HEAD)
```

5. Отменить уже опубликованные коммиты с помощью новых коммитов

Воспользуйтесь командой `git revert`. Она создает новые коммиты, по одному на каждый отменяемый коммит. Таким образом, если нужно отменить все коммиты после `aaaaaa`:

```
# можно перечислить отменяемые коммиты
git revert bbbbbb cccccc dddddd

# можно задать диапазон от более раннего к более позднему (новому)
git revert bbbbbb..dddddd

# либо в относительных ссылках
git revert HEAD~2..HEAD

# можно отменить коммит слияния, указывая явным образом номер предка (в нашем
примере таких нет):
git revert -m 1 abcdef

# после этого подтвердите изменения:
git commit -m'детальное описание, что и почему сделано'
```

> **Восстановление**: Если revert-коммит оказался ошибочным, используйте этот ответ.

Поделиться Улучшить ответ Отслеживать

изменён 5 окт 2017 в 13:16

ответ дан 23 июн 2015 в 9:43

Nick Volynkin ♦

32.6k ● 24 ● 124 ● 211

16 Крутой ответ! Ещё бывает полезно использовать *git worktree* вместо *stash*. – 0andriy 8 авг 2017 в 6:17

4 todo: добавить параграф про `git worktree`. – Nick Volynkin ♦ 23 сен 2017 в 3:36

А где чери пик ? – Shwarz Andrei 5 окт 2017 в 19:13

@ShwarzAndrei а как он относится к откатыванию? – Nick Volynkin ♦ 6 окт 2017 в 0:39