

УДК 004.4
ББК 32.973.2-018.2
С28

*Книгу «Git tips and tricks» на английском языке можно найти здесь:
<https://opensource.com/downloads/git-tricks-tips>.*

Она распространяется по лицензии

Creative Commons Attribution-ShareAlike 4.0.

*В книге представлены статьи, написанные Брентом Ластером,
Радживом Берой, Сетом Кенлоном, Олафом Альдерсом,
Рамакришной Паттнаиком, Агилом Антониєм и Ноа Барки.*

Секреты Git. Никогда не теряй свой код! — Москва : Издатель-
ство АСТ, 2024. — 96 с. : ил. — (Учимся программировать).
ISBN 978-5-17-160269-7.

Данное издание может стать незаменимым помощником для любого пользователя системы контроля версий Git, в наши дни широко распространенной в сфере разработки программного обеспечения благодаря своей «продвинутой» и универсальности. Авторы разделов этой книги — опытные разработчики, постоянно и активно использующие Git в решении повседневных задач при написании и систематизации программного кода. В частности, здесь рассмотрены восстановление предыдущих состояний системы в случае неудачных действий с Git, возможность отправки в удаленный репозиторий выборочных коммитов, особенности быстрого переключения с одной задачи на другую без потери прогресса в работе, использование рабочих деревьев, восстановление удаленного репозитория и многое другое.

УДК 004.4

ББК 32.973.2-018.2

ISBN 978-5-17-160269-7

Перевод на русский язык: ООО «Интеджер».
Издание на русском языке: ООО «Издательство АСТ».

Содержание

Как сбрасывать, отменять и “откатываться” к предыдущим состояниям в Git	8
Как сбросить фиксацию в Git.....	8
Как отменить коммит в Git.....	12
Команда rebase (перемещение).....	17
Что такое Git cherry-picking?	23
Что такое cherry-pick?	23
Преимущества cherry-pick	24
Использование команды cherry-pick	25
Попробуйте сами	27
Причины, по которым следует избегать команды cherry-pick.....	29
Поиск изменений в коммите Git.....	30
Как определить, какой именно файл в коммите был изменен	30

Команда git whatchanged	33
Просмотр изменений.....	34
Простые команды для получения сложных результатов.....	35
Свободно экспериментируйте над своим кодом с помощью Git worktree	37
Что такое рабочее дерево Git.....	38
Вывод списка активных рабочих деревьев	41
Перемещение рабочего дерева	41
Удаление рабочего дерева	42
Когда использовать рабочие деревья	42
Четыре совета по переключению контекста в Git.....	44
Решение № 1: “тайник” + ветка.....	45
Решение № 2: коммит WIP + ветка.....	46
Решение № 3: создать новый клон репозитория....	47
Решение № 4: git worktree	48
Еще несколько дополнительных советов.....	51
Важные замечания.....	51
git rev-parse	52
Выберите наиболее подходящий метод для ваших задач	53

Практическое руководство по использованию команды git stash	54
Почему так важна команда git stash?	55
Как использовать git stash.....	56
Как создать “тайник”	57
Вывод на экран записей, хранящихся в ваших “тайниках”	59
Извлечение сохраненных изменений.....	60
Удаление “тайника”	61
Проверка диффов “тайников”	61
Переключение на новую ветку.....	62
Прятанье, которое не изменяет журнал ссылок (reflog).....	63
Как переименовать ветку, удалить ветку и найти автора ветки в Git	65
Переименование ветки с помощью Git.....	66
Удаление локальных и удаленных веток с помощью Git	67
Поиск автора удаленной ветки средствами Git.....	69
Обучайтесь ветвлению в Git.....	71
Удаление локальной ссылки на удаленную ветку в Git	72
Управление вашим Git-репозиторием	74

Мое руководство по безопасному использованию команды <code>git push</code>	75
Команда <code>git push</code>	76
Правило “быстрой перемотки”	77
Когда можно использовать параметр <code>--force</code>	77
Простой сценарий	79
Альтернатива: <code>push --force-with-lease</code>	80
Руководство: как справиться с деструктивным <code>--force</code>	81
1. Вы были последним, кто выкладывал информацию в ветку перед ошибкой?	81
2. Я случайно использовал <code>push --force</code> в своем репозитории и хочу вернуться к предыдущей версии. Что мне делать?	82
Общее восстановление	85
3. Восстановление удаленной ветки <code>push --force</code> с помощью <code>git fsck</code>	85
Защищенные ветви и обзоры кода.....	86
Резюме.....	87
Восстановление после неудачного перемещения ветки (<code>git rebase</code>) с помощью команды <code>git reflog</code>	88
Git <code>reflog</code>	88
Просмотрите свою историю с помощью <code>reflog</code>	89
Найдите последний хороший коммит	90

Восстановите коммит 90

Контроль версий Git..... 91

Загляните в репозиторий Git

с помощью команды rev-parse 92

Получение каталога верхнего уровня..... 93

Вернуться домой 94

Текущее местоположение 95

Скрипты Git 95

Как сбрасывать, отменять и “откатываться” к предыдущим состояниям в Git

Брент Ластер

Один из наименее понятных (и наиболее важных) аспектов работы с Git — это умение легко “откатиться” к тому состоянию системы контроля версий, на котором вы остановились перед внесением изменений, то есть аспект, связанный с тем, как можно легко отменить даже значительные изменения, внесенные в репозиторий. В этой главе будет рассказано о том, как можно просто и элегантно сбросить, отменить и полностью “откатиться” к предыдущим состояниям с помощью отдельных команд Git.

Как сбросить фиксацию в Git

Начнем с команды `Git reset`. Практически вы можете считать ее “откатом” — она “откатывает” указатель

так, что он указывает на ваше локальное окружение для предыдущего коммита. Под локальным окружением подразумеваются ваш локальный репозиторий, область подготовленных для коммита файлов и рабочий каталог.

Посмотрите на рисунок 1. Здесь изображена серия коммитов в Git. Ветка в Git — это просто поименованный перемещаемый указатель на конкретный коммит. В данном случае ветка *master* — это указатель на последний коммит в цепочке.

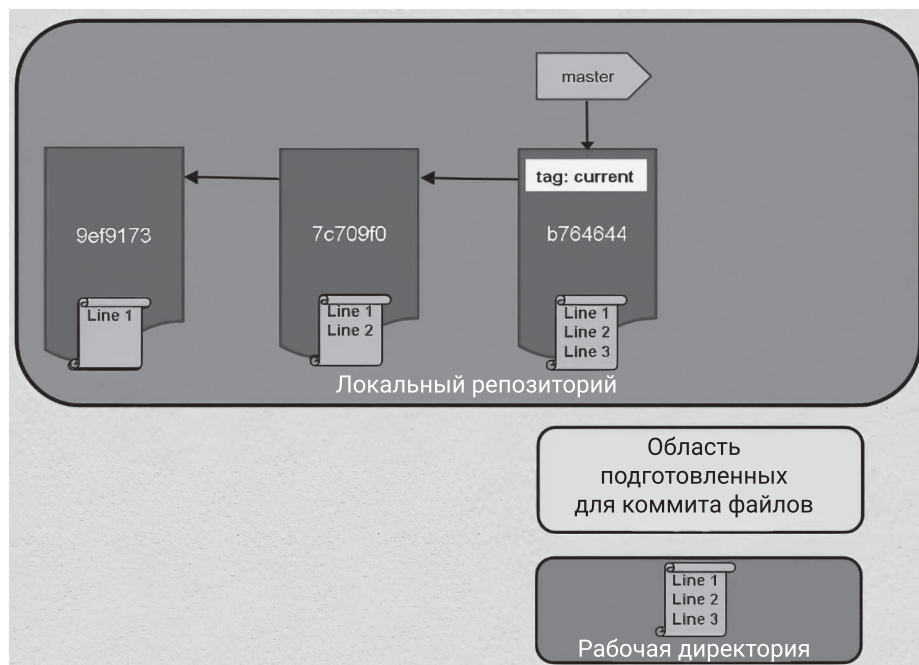


Рис. 1
(Brent Laster, CC BY-SA 4.0)

Если вы посмотрите на то, что сейчас находится в вашей мастер-ветке, вы увидите следующую цепочку коммитов, сделанных на данный момент:

10 Как сбрасывать, отменять и “откатываться” к предыдущим...

```
$ git log --oneline
```

```
b764644 File with three lines // Файл с тремя строками
```

```
7c709f0 File with two lines // Файл с двумя строками
```

```
9ef9173 File with one line // Файл с одной строкой
```

Что делать, если вы хотите “откатиться” к предыдущему коммиту? Все просто — вы можете переместить указатель ветки. Git предоставляет команду `reset`, чтобы сделать это за вас. Например, если вы хотите сбросить указатель *master* так, чтобы он указывал на коммит, находящийся на два шага назад от текущего коммита, вы можете воспользоваться одним из следующих способов:

```
$ git reset 9ef9173 (используя абсолютное значение хеша SHA1 коммита 9ef9173)
```

или

```
$ git reset current~2 (используя относительное значение -2 перед тегом “current”)
```

На рисунке 2 показаны результаты этой операции. После этого, если выполнить команду `git log` в текущей ветке (*master*), мы увидим только один коммит:

```
$ git log --oneline
```

```
9ef9173 File with one line // Файл с одной строкой
```

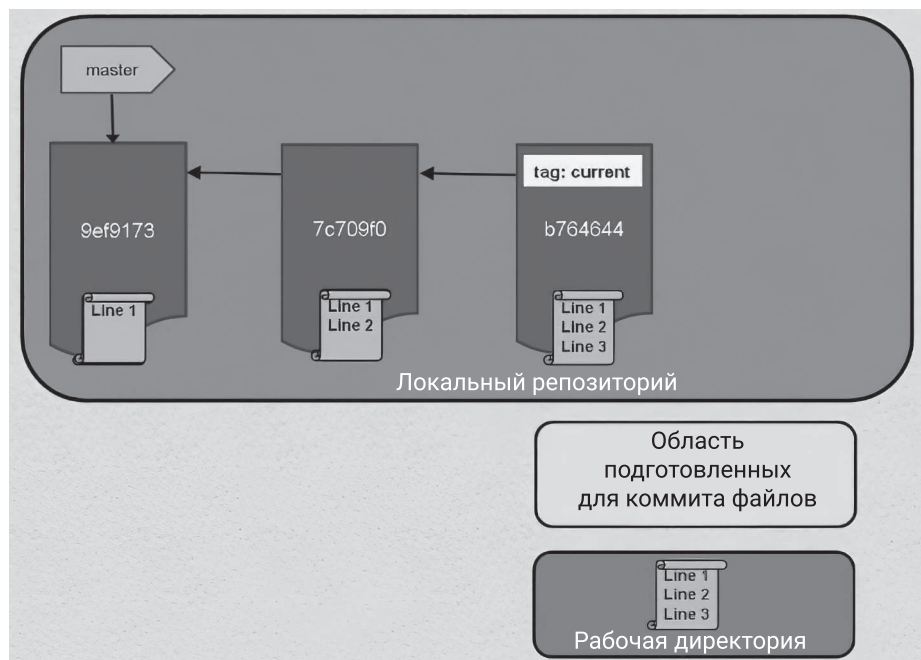


Рис. 2
(Brent Laster, CC BY-SA 4.0)

Команда `git reset` также содержит параметры для обновления других частей вашего локального окружения (область подготовленных для коммита файлов и рабочей директории) содержимым коммита, на котором вы сейчас остановились. Это такие параметры, как: `hard` — жесткий сброс коммита, на который указывал указатель до сброса, с физическим удалением информации из рабочего каталога и области индексирования; `soft` — сброс только указателя в репозитории (аналог команды `git commit --amend`); `mixed` (аналогичное поведение предлагается по умолчанию, то есть в ситуации, когда параметры отсутствуют) — сброс указателя и сброс области индексирования (удаление изменений в индексе для самого