

?????? ?????????????? ??????

Следующие два раздела демонстрируют как работать с индексом и изменениями в рабочем каталоге. Радует, что команда, которой вы определяете состояние этих областей, также подсказывает вам как отменять изменения в них. Например, вы изменили два файла и хотите добавить их в разные коммиты, но случайно выполнили команду `git add *` и добавили в индекс оба. Как исключить из индекса один из них? Команда `git status` напомнит вам:

```
$ git add *
$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

    renamed:    README.md -> README
    modified:   CONTRIBUTING.md
```

Прямо под текстом «Changes to be committed» говорится: используйте `git reset HEAD <file>...` для исключения из индекса. Давайте последуем этому совету и отменим индексирование файла `CONTRIBUTING.md`:

```
$ git reset HEAD CONTRIBUTING.md
Unstaged changes after reset:
M CONTRIBUTING.md
$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

    renamed:    README.md -> README

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

    modified:   CONTRIBUTING.md
```

Команда выглядит несколько странно, но — работает! Файл `CONTRIBUTING.md` изменен, но больше не добавлен в индекс.

Команда `git reset` может быть опасной если вызвать её с параметром `--hard`. В приведённом примере файл не был затронут, следовательно команда относительно безопасна.

На текущий момент этот магический вызов — всё, что вам нужно знать о команде `git reset`. Мы рассмотрим в деталях что именно делает `reset` и как с её помощью делать действительно интересные вещи в разделе [Раскрытие тайн reset](#) главы 7.

?????? ??????????? ? ???????

Что делать, если вы поняли, что не хотите сохранять свои изменения файла `CONTRIBUTING.md`? Как можно просто отменить изменения в нём — вернуть к тому состоянию, которое было в последнем коммите (или к начальному после клонирования, или ещё как-то полученному)? Нам повезло, что `git status` подсказывает и это тоже.

В выводе команды из последнего примера список изменений выглядит примерно так:

```
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

   modified:   CONTRIBUTING.md
```

Здесь явно сказано как отменить существующие изменения. Давайте так и сделаем:

```
$ git checkout -- CONTRIBUTING.md
$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

   renamed:   README.md -> README
```

Как видите, откат изменений выполнен.

Важно понимать, что `git checkout -- <file>` — опасная команда. Все локальные изменения в файле пропадут — Git просто заменит его версией из последнего коммита. Ни в коем случае не используйте эту команду, если вы не уверены, что изменения в файле вам не нужны.

Если вы хотите сохранить изменения в файле, но прямо сейчас их нужно отменить, то есть способы получше, такие как ветвление и припрятывание — мы рассмотрим их в главе

[Ветвление в Git.](#)

Помните, всё что попало в *коммит* почти всегда Git может восстановить. Можно восстановить даже коммиты из веток, которые были удалены, или коммиты, перезаписанные параметром `--amend` (см. [Восстановление данных](#)). Но всё, что не было включено в коммит и потеряно — скорее всего, потеряно навсегда.

?????? ?????????? ? ?????????? git restore

Git версии 2.23.0 представил новую команду: `git restore`. По сути, это альтернатива `git reset`, которую мы только что рассмотрели. Начиная с версии 2.23.0, Git будет использовать `git restore` вместо `git reset` для многих операций отмены.

Давайте проследим наши шаги и отменим действия с помощью `git restore` вместо `git reset`.

?????? ?????????????? ?????? ? ?????????? git restore

В следующих двух разделах показано, как работать с индексом и изменениями рабочей копии с помощью `git restore`. Приятно то, что команда, которую вы используете для определения состояния этих двух областей, также напоминает вам, как отменить изменения в них. Например, предположим, что вы изменили два файла и хотите зафиксировать их как два отдельных изменения, но случайно набираете `git add *` и индексируете их оба. Как вы можете убрать из индекса один из двух? Команда `git status` напоминает вам:

```
$ git add *
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
   modified:   CONTRIBUTING.md
   renamed:   README.md -> README
```

Прямо под текстом «Changes to be committed», написано использовать `git restore --staged <file> ...` для отмены индексации файла. Итак, давайте воспользуемся этим советом, чтобы убрать из индекса файл `CONTRIBUTING.md`:

```
$ git restore --staged CONTRIBUTING.md
$ git status
On branch master
Changes to be committed:
```

```
(use "git restore --staged <file>..." to unstage)
```

```
□renamed: README.md -> README
```

```
Changes not staged for commit:
```

```
(use "git add <file>..." to update what will be committed)
```

```
(use "git restore <file>..." to discard changes in working directory)
```

```
□modified: CONTRIBUTING.md
```

Файл `CONTRIBUTING.md` изменен, но снова не индексируется.

????? ?????????????? ?????? ? ?????????? git restore

Что, если вы поймете, что не хотите сохранять изменения в файле `CONTRIBUTING.md`? Как легко его откатить — вернуть обратно к тому, как он выглядел при последнем коммите (или изначально клонирован, или каким-либо образом помещён в рабочий каталог)? К счастью, `git status` тоже говорит, как это сделать. В выводе последнего примера, неиндексированная область выглядит следующим образом:

```
Changes not staged for commit:
```

```
(use "git add <file>..." to update what will be committed)
```

```
(use "git restore <file>..." to discard changes in working directory)
```

```
□modified: CONTRIBUTING.md
```

Он довольно недвусмысленно говорит, как отменить сделанные вами изменения. Давайте сделаем то, что написано:

```
$ git restore CONTRIBUTING.md
```

```
$ git status
```

```
On branch master
```

```
Changes to be committed:
```

```
(use "git restore --staged <file>..." to unstage)
```

```
□renamed: README.md -> README
```

Важно понимать, что `git restore <file>` — опасная команда. Любые локальные изменения, внесённые в этот файл, исчезнут — Git просто заменит файл последней зафиксированной версией. Никогда не используйте эту команду, если точно не знаете, нужны ли вам эти несохранённые локальные изменения.

Revision #2

Created 2024-05-02 06:08:54 UTC by Антон Сергеевич Абраменко

Updated 2024-05-02 06:31:27 UTC by Антон Сергеевич Абраменко