ПОЛУЧЕНИЕ СТАРЫХ ВЕРСИЙ

Git позволяет очень просто путешествовать во времени, по крайней мере, для вашего проекта. Команда *checkout* обновит вашу рабочую директорию до любого предыдущего коммита.

1. Получите хеши предыдущих коммитов

Выполните

```
git log
```

Результат

```
$ git log
b7614c1 2023-11-28 | Added HTML header (HEAD -> main) [Alexander Shvets]
46afaff 2023-11-28 | Added standard HTML page tags [Alexander Shvets]
78433de 2023-11-28 | Added h1 tag [Alexander Shvets]
5836970 2023-11-28 | Initial commit [Alexander Shvets]
```

Просмотрите историю изменений и найдите хеш первого коммита. Он должен быть в последней строке результата *git log*. Используйте этот хеш (достаточно первых 7 символов) в команде ниже. Затем проверьте содержимое файла *hello.html.*

Выполните

```
git checkout <hash>
cat hello.html
```

Многие команды *Git* принимают хеши коммитов в качестве аргументов. Хеши коммитов будут отличаться в разных репозиториях, поэтому когда вы видите, что в команде есть пометка <*hash*>, то это значит, что вам надо подставить вместо нее реальный хеш из вашего репозитория.

Результат

```
$ git checkout 5836970
Note: switching to '5836970'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by switching back to a branch.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -c with the switch command. Example:

  git switch -c <new-branch-name>

Or undo this operation with:

  git switch -

Turn off this advice by setting config variable advice.detachedHead to false

HEAD is now at 5836970 Initial commit
$ cat hello.html
Hello, World!
```

Обратите внимание, что сейчас содержимое файла hello.html — это тот самый текст, с которого мы начинали.

2.* Вернитесь к последней версии в ветке *main

Чтобы вернуться к последней версии нашего кода, нам нужно переключиться на ветку по умолчанию, *main*. Для переключения между ветками можно воспользоваться командой *switch*.

Команда *checkout* в течение длительного времени была своеобразным швейцарским ножом в мире *Git*. Она имеет множество различных опций, которые позволяют выполнять совершенно разные вещи: переключать ветки, сбрасывать код и так далее. В какой-то момент команда *Git* решила разделить ее на несколько команд. Команда *switch* является одной из них — ее единственным назначением является переключение между ветками. Команда *checkout* все еще доступна, но использовать ее для переключения веток больше не рекомендуется.

Выполните

```
git switch main
cat hello.html
```

Результат

```
$ git switch main
Previous HEAD position was 5836970 Initial commit
Switched to branch 'main'
$ cat hello.html
<html>
  <head>
  </head>
  <body>
    <h1>Hello, World!</h1>
  </body>
</html>
```

main — имя ветки по умолчанию. Переключаясь на ветку, вы попадаете на её последнюю версию.

СОЗДАНИЕ ТЕГОВ ВЕРСИЙ

Думаю, вы согласитесь, что работать с хешами коммитов напрямую просто неудобно. Можно обозначать конкретные коммиты понятными для человека названиями. Таким образом, можно четко видеть важные вехи в истории проекта. Кроме того, вы могли бы легко перейти к определенной версии проекта по ее названию. Именно для этого в *Git* придумали теги.

Давайте назовем текущую версию страницы *hello.html* первой, то есть *v1*.

Выполните

```
git tag v1
git log
```

Результат

```
$ git tag v1
$ git log
b7614c1 2023-11-28 | Added HTML header (HEAD -> main, tag: v1) [Alexander Shvets]
46afaff 2023-11-28 | Added standard HTML page tags [Alexander Shvets]
78433de 2023-11-28 | Added h1 tag [Alexander Shvets]
5836970 2023-11-28 | Initial commit [Alexander Shvets]
```

Теперь текущая версия страницы называется v1.

2. Теги для предыдущих версий

Обозначим версию, предшествующую текущей, названием *v1-beta*. Прежде всего, мы переключимся на предыдущую версию. Вместо того чтобы искать хеш коммита, мы будем использовать обозначение **^**, а именно ***v1^***, указывающее на коммит, предшествующий *v1*.

Если обозначение ***v1^*** вызывает у вас какие-то проблемы, попробуйте также ***v1~1***, указывающее на ту же версию. Это обозначение можно определить как «первую версию, предшествующую ***v1***».

Выполните

```
git checkout v1^
cat hello.html
```

Результат

```
$ git checkout v1^
Note: switching to 'v1^'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by switching back to a branch.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -c with the switch command. Example:

  git switch -c <new-branch-name>

Or undo this operation with:

  git switch -

Turn off this advice by setting config variable advice.detachedHead to false

HEAD is now at 46afaff Added standard HTML page tags
$ cat hello.html
<html>
  <body>
    <h1>Hello, World!</h1>
  </body>
</html>
```

Это версия с тегами <html> и <body>, но еще *пока* без <head>. Давайте сделаем ее версией v1-beta.

Выполните

```
git tag v1-beta
git log
```

Результат

```
$ git tag v1-beta
$ git log
46afaff 2023-11-28 | Added standard HTML page tags (HEAD, tag: v1-beta) [Alexander Shvets]
78433de 2023-11-28 | Added h1 tag [Alexander Shvets]
5836970 2023-11-28 | Initial commit [Alexander Shvets]
```

3.Переключение по имени тега

Теперь попробуйте попереключаться между двумя отмеченными версиями.

Выполните

```
git checkout v1
git checkout v1-beta
```

Результат

```
$ git checkout v1
Previous HEAD position was 46afaff Added standard HTML page tags
HEAD is now at b7614c1 Added HTML header
$ git checkout v1-beta
Previous HEAD position was b7614c1 Added HTML header
HEAD is now at 46afaff Added standard HTML page tags
```

4. Просмотр тегов с помощью команды tag

Вы можете увидеть, какие теги доступны, используя команду git tag.

Выполните

```
git tag
```

Результат

```
$ git tag
v1
v1-beta
```

5. Просмотр Тегов в логах

Вы также можете посмотреть теги в логе.

Выполните

```
git log main --all
```

Результат

```
$ git log main --all
b7614c1 2023-11-28 | Added HTML header (tag: v1, main) [Alexander Shvets]
46afaff 2023-11-28 | Added standard HTML page tags (HEAD, tag: v1-beta) [Alexander Shvets]
78433de 2023-11-28 | Added h1 tag [Alexander Shvets]
5836970 2023-11-28 | Initial commit [Alexander Shvets]
```

Вы можете видеть теги (v1 и v1-beta) в логе вместе с именем ветки (main). Кроме того, метка HEAD показывает коммит, на который вы переключились (на данный момент это v1-beta)

ОТМЕНА ЛОКАЛЬНЫХ ИЗМЕНЕНИЙ (ДО ИНДЕКСАЦИИ)

1. Переключитесь на ветку main

Убедитесь, что вы находитесь на последнем коммите ветки main, прежде чем продолжить работу.

Выполните

```
git switch main
```

2. Измените hello.html

Иногда после того как вы изменили файл в рабочей директории, вы передумали и хотите просто вернуться к тому, что уже было закоммичено. Команда checkout справится с этой задачей.

Внесите изменение в файл hello.html в виде нежелательного комментария.

```
<html>
  <head>
  </head>
  <body>
    <h1>Hello, World!</h1>
    <!-- This is a bad comment. We want to revert it. -->
  </body>
</html>
```

3. Проверьте состояние

Сначала проверьте состояние рабочей директории.

Выполните

```
git status
```

Результат

```
$ git status
On branch main
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:   hello.html

no changes added to commit (use "git add" and/or "git commit -a")
```

Мы видим, что файл hello.html был изменен, но еще не проиндексирован.

4. Отмена изменений в рабочем каталоге

Используйте команду checkout для переключения в версию файла hello.html в репозитории.

Выполните

```
git checkout hello.html
git status
cat hello.html
```

Результат

```
$ git checkout hello.html
Updated 1 path from the index
$ git status
On branch main
nothing to commit, working tree clean
$ cat hello.html
<html>
  <head>
  </head>
  <body>
    <h1>Hello, World!</h1>
  </body>
</html>
```

Команда status показывает нам, что в рабочей директории не было сделано никаких незафиксированных изменений. И «нежелательный комментарий» больше не является частью содержимого файла.

ОТМЕНА ПРОИНДЕКСИРОВАННЫХ ИЗМЕНЕНИЙ (ПЕРЕД КОММИТОМ)

1. Измените файл и проиндексируйте изменения

Внесите изменение в файл hello.html в виде нежелательного комментария

```
<html>
  <head>
    <!-- This is an unwanted but staged comment -->
  </head>
  <body>
    <h1>Hello, World!</h1>
  </body>
</html>
```

Проиндексируйте это изменение.

Выполните

```
git add hello.html
```

2. Проверьте состояние

Проверьте состояние нежелательного изменения.

Выполните

```
git status
```

Результат

```
$ git status
On branch main
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    modified:   hello.html
```

Состояние показывает, что изменение было проиндексировано и готово к коммиту.

3. Сбросьте области подготовки

Команда reset сбрасывает область подготовки к HEAD. Это очищает область подготовки от изменений, которые мы только что проиндексировали.

Выполните

```
git reset HEAD hello.html
```

Результат

```
$ git reset HEAD hello.html
Unstaged changes after reset:
M   hello.html
```

Команда reset (по умолчанию) не изменяет рабочую директорию. Поэтому рабочая директория всё еще содержит нежелательный комментарий. Мы можем использовать команду checkout из предыдущего урока, чтобы убрать нежелательные изменения в рабочей директории.

4. Переключитесь на версию коммита

Выполните

```
git checkout hello.html
git status
```

Результат

```
$ git checkout hello.html
Updated 1 path from the index
$ git status
On branch main
nothing to commit, working tree clean
```

Наша рабочая директория опять чиста.

УДАЛЕНИЕ КОММИТОВ ИЗ ВЕТКИ

revert из предыдущего раздела является мощной командой, которая позволяет отменить любые коммиты в репозиторий. Однако, и оригинальный и «отмененный» коммиты видны в истории ветки (при использовании команды git log).

Часто мы делаем коммит, и сразу понимаем, что это была ошибка. Было бы неплохо иметь команду «возврата», которая позволила бы нам сделать вид, что неправильного коммита никогда и не было. Команда «возврата» даже предотвратила бы появление нежелательного коммита в истории git log.

1. Команда reset

Мы уже видели команду reset и использовали ее для согласования области подготовки с выбранным коммитом (в предыдущем уроке мы использовали коммит HEAD).

Если выполнить команду reset с указанием ссылки на коммит (т.е. метки HEAD, имени ветки или тега, хеша коммита), то команда...

1. Изменит текущую ветку, чтобы она указывала на указанный коммит.
2. Опционально сбросит область подготовки до соответствия с указанным коммитом.
3. Опционально сбросит рабочую директорию до соответствия с указанным коммитом.

2. **Проверьте нашу историю**

Давайте сделаем быструю проверку нашей истории коммитов.

Выполните

```
git log
```

Результат

```
$ git log
86364a1 2023-11-28 | Revert "Oops, we didn't want this commit" (HEAD -> main) [Alexander Shvets]
6a44bec 2023-11-28 | Oops, we didn't want this commit [Alexander Shvets]
b7614c1 2023-11-28 | Added HTML header (tag: v1) [Alexander Shvets]
46afaff 2023-11-28 | Added standard HTML page tags (tag: v1-beta) [Alexander Shvets]
78433de 2023-11-28 | Added h1 tag [Alexander Shvets]
5836970 2023-11-28 | Initial commit [Alexander Shvets]
```

Мы видим, что два последних коммита в этой ветке - «Oops» и «Revert Oops». Давайте удалим их с помощью сброса.

3. Для начала отметьте эту ветку

Но прежде чем удалить коммиты, давайте отметим последний коммит тегом, чтобы потом можно было его найти.

Выполните

```
git tag oops
```

Результат

```
$ git log
86364a1 2023-11-28 | Revert "Oops, we didn't want this commit" (HEAD -> main, tag: oops) [Alexander Shvet
6a44bec 2023-11-28 | Oops, we didn't want this commit [Alexander Shvets]
b7614c1 2023-11-28 | Added HTML header (tag: v1) [Alexander Shvets]
46afaff 2023-11-28 | Added standard HTML page tags (tag: v1-beta) [Alexander Shvets]
78433de 2023-11-28 | Added h1 tag [Alexander Shvets]
5836970 2023-11-28 | Initial commit [Alexander Shvets]
```

4. Сброс к коммиту, предшествующему oops

Глядя на историю лога видно, что коммит с тегом v1 является коммитом, предшествующим ошибочному коммиту. Давайте сбросим ветку до этой точки. Поскольку ветка имеет тег, мы можем использовать имя тега в команде сброса reset (если не имеет тега, можно использовать хеш коммита).

Выполните

```
git reset --hard v1
git log
```

Результат

```
$ git reset --hard v1
HEAD is now at b7614c1 Added HTML header
$ git log
b7614c1 2023-11-28 | Added HTML header (HEAD -> main, tag: v1) [Alexander Shvets]
46afaff 2023-11-28 | Added standard HTML page tags (tag: v1-beta) [Alexander Shvets]
78433de 2023-11-28 | Added h1 tag [Alexander Shvets]
5836970 2023-11-28 | Initial commit [Alexander Shvets]
```

Ветка main теперь указывает на коммит v1, а коммитов "Revert Oops" и "Oops" в ветке уже нет. Параметр --hard указывает, что рабочая директория должна быть приведена к тому состоянию, которое соответствует HEAD-коммиту ветки.

5. Ничего никогда не теряется

Что же случается с ошибочными коммитами? Оказывается, что коммиты все еще находятся в репозитории. На самом деле, мы все еще можем на них ссылаться. Помните, в начале этого урока мы создали для отмененного коммита тег oops? Давайте посмотрим на *все* коммиты.

Выполните

```
git log --all
```

Результат

```
$ git log --all
b7614c1 2023-11-28 | Added HTML header (HEAD -> main, tag: v1) [Alexander Shvets]
86364a1 2023-11-28 | Revert "Oops, we didn't want this commit" (tag: oops) [Alexander Shvets]
46afaff 2023-11-28 | Added standard HTML page tags (tag: v1-beta) [Alexander Shvets]
6a44bec 2023-11-28 | Oops, we didn't want this commit [Alexander Shvets]
78433de 2023-11-28 | Added h1 tag [Alexander Shvets]
5836970 2023-11-28 | Initial commit [Alexander Shvets]
```

Мы видим, что ошибочные коммиты не исчезли. Они все еще находятся в репозитории. Просто они отсутствуют в ветке main. Если бы мы не отметили их тегами, они по-прежнему находились бы в репозитории, но не было бы никакой возможности ссылаться на них, кроме как при помощи хешей этих коммитов. Коммиты, на которые нет ссылок, остаются в репозитории до тех пор, пока не будет запущен сборщик мусора.

6. Опасность сброса

Сброс в локальных ветках, как правило, безопасен. Последствия любой «аварии» как правило, можно восстановить простым сбросом с помощью нужного коммита.

Однако, если ветка уже стала общедоступной на удаленных репозиториях, сброс может сбить с толку других пользователей ветки.

УДАЛЕНИЕ ТЕГА Oops

1. Удаление тега oops

Тег oops свою функцию выполнил, давайте удалим его. Это позволит внутреннему механизму Git убрать остаточные коммиты, на которые теперь не ссылаются никакие ветки или теги.

Выполните

```
git tag -d oops
git log --all
```

Результат

```
$ git tag -d oops
Deleted tag 'oops' (was 86364a1)
$ git log --all
b7614c1 2023-11-28 | Added HTML header (HEAD -> main, tag: v1) [Alexander Shvets]
46afaff 2023-11-28 | Added standard HTML page tags (tag: v1-beta) [Alexander Shvets]
78433de 2023-11-28 | Added h1 tag [Alexander Shvets]
5836970 2023-11-28 | Initial commit [Alexander Shvets]
```

ВНЕСЕНИЕ ИЗМЕНЕНИЙ В КОММИТЫ

1. **Измените страницу, а затем сделайте коммит**

Добавьте в страницу комментарий автора.**Файл: hello.html**

```
<!-- Author: Alexander Shvets -->
<html>
  <head>
  </head>
  <body>
    <h1>Hello, World!</h1>
  </body>
</html>
```

Выполните

```
git add hello.html
git commit -m "Added copyright statement"
git log
```

Результат

```
$ git add hello.html
$ git commit -m "Added copyright statement"
[main e641c0e] Added copyright statement
 1 file changed, 1 insertion(+)
$ git log
e641c0e 2023-11-28 | Added copyright statement (HEAD -> main) [Alexander Shvets]
b7614c1 2023-11-28 | Added HTML header (tag: v1) [Alexander Shvets]
46afaff 2023-11-28 | Added standard HTML page tags (tag: v1-beta) [Alexander Shvets]
78433de 2023-11-28 | Added h1 tag [Alexander Shvets]
5836970 2023-11-28 | Initial commit [Alexander Shvets]
```

2. Ой... необходим email

Однако после создания коммита вы понимаете, что любой хороший комментарий должен включать электронную почту автора. Обновите страницу hello.html, включив в нее email.

Файл: hello.html

```
<!-- Author: Alexander Shvets (alex@githowto.com) -->
<html>
  <head>
  </head>
  <body>
    <h1>Hello, World!</h1>
  </body>
</html>
```

***03*Измените предыдущий коммит**

Мы действительно не хотим создавать отдельный коммит только ради электронной почты. Давайте изменим предыдущий коммит, включив в него адрес электронной почты.

Выполните

```
git add hello.html
git commit --amend -m "Added copyright statement with email"
```

Результат

```
$ git add hello.html
$ git commit --amend -m "Added copyright statement with email"
[main 9288a33] Added copyright statement with email
 Date: Tue Nov 28 05:51:38 2023 -0600
 1 file changed, 1 insertion(+)
```

4. Просмотр истории

Выполните

```
git log
```

Результат

```
$ git log
9288a33 2023-11-28 | Added copyright statement with email (HEAD -> main) [Alexander Shvets]
b7614c1 2023-11-28 | Added HTML header (tag: v1) [Alexander Shvets]
46afaff 2023-11-28 | Added standard HTML page tags (tag: v1-beta) [Alexander Shvets]
78433de 2023-11-28 | Added h1 tag [Alexander Shvets]
5836970 2023-11-28 | Initial commit [Alexander Shvets]
```

Мы можем увидеть, что оригинальный коммит «автор» заменен коммитом «автор/email». Этого же эффекта можно достичь путем сброса последнего коммита в ветке, и повторного коммита новых изменений.