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

Как и большинство других систем контроля версий, Git имеет возможность помечать определённые моменты в истории как важные. Как правило, эта функциональность используется для отметки моментов выпуска версий (v1.0, и т. п.). Такие пометки в Git называются тегами. В этом разделе вы узнаете, как посмотреть имеющиеся теги, как создать новые или удалить существующие, а также какие типы тегов существуют в Git.

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

Просмотреть список имеющихся тегов в Git можно очень просто. Достаточно набрать команду `git tag` (параметры `-l` и `--list` опциональны):

```
$ git tag
v1.0
v2.0
```

Данная команда перечисляет теги в алфавитном порядке; порядок их отображения не имеет существенного значения.

Так же можно выполнить поиск тега по шаблону. Например, репозиторий Git содержит более 500 тегов. Если вы хотите посмотреть теги выпусков 1.8.5, то выполните следующую команду:

```
$ git tag -l "v1.8.5*"
v1.8.5
v1.8.5-rc0
v1.8.5-rc1
v1.8.5-rc2
v1.8.5-rc3
v1.8.5.1
v1.8.5.2
v1.8.5.3
v1.8.5.4
v1.8.5.5
```

> Для отображение тегов согласно шаблону требуются параметры `-l` или `--list`. Если вы хотите посмотреть весь список тегов, запуск команды `git tag` неявно

> подразумевает это и выводит полный список; использование параметров `-l` или `--list` в этом случае опционально. Если вы хотите отфильтровать список тегов согласно шаблону, использование параметров `-l` или `--list` становится обязательным.

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

Git использует два основных типа тегов: легковесные и аннотированные.

Легковесный тег — это что-то очень похожее на ветку, которая не изменяется — просто указатель на определённый коммит.

А вот аннотированные теги хранятся в базе данных Git как полноценные объекты. Они имеют контрольную сумму, содержат имя автора, его e-mail и дату создания, имеют комментарий и могут быть подписаны и проверены с помощью GNU Privacy Guard (GPG). Обычно рекомендуется создавать аннотированные теги, чтобы иметь всю перечисленную информацию; но если вы хотите сделать временную метку или по какой-то причине не хотите сохранять остальную информацию, то для этого годятся и легковесные.

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

Создание аннотированного тега в Git выполняется легко. Самый простой способ — это указать `-a` при выполнении команды `tag`:

```
$ git tag -a v1.4 -m "my version 1.4"
$ git tag
v0.1
v1.3
v1.4
```

Опция `-m` задаёт сообщение, которое будет храниться вместе с тегом. Если не указать сообщение, то Git запустит редактор, чтобы вы смогли его ввести.

С помощью команды `git show` вы можете посмотреть данные тега вместе с коммитом:

```
$ git show v1.4
tag v1.4
Tagger: Ben Straub <ben@straub.cc>
Date:   Sat May 3 20:19:12 2014 -0700

my version 1.4

commit ca82a6dff817ec66f44342007202690a93763949
```

```
Author: Scott Chacon <schacon@gee-mail.com>
Date:   Mon Mar 17 21:52:11 2008 -0700

    Change version number
```

Здесь приведена информация об авторе тега, дате его создания и аннотирующее сообщение перед информацией о коммите.

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

Легковесный тег — это ещё один способ пометить коммит. По сути, это контрольная сумма коммита, сохранённая в файл — больше никакой информации не хранится. Для создания легковесного тега не передавайте опций `-a`, `-s` и `-m`, укажите только название:

```
$ git tag v1.4-lw
$ git tag
v0.1
v1.3
v1.4
v1.4-lw
v1.5
```

На этот раз при выполнении `git show` для этого тега вы не увидите дополнительной информации. Команда просто покажет коммит:

```
$ git show v1.4-lw
commit ca82a6dff817ec66f44342007202690a93763949
Author: Scott Chacon <schacon@gee-mail.com>
Date:   Mon Mar 17 21:52:11 2008 -0700

    Change version number
```

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

Также возможно помечать уже пройденные коммиты. Предположим, история коммитов выглядит следующим образом:

```
$ git log --pretty=oneline
15027957951b64cf874c3557a0f3547bd83b3ff6 Merge branch 'experiment'
a6b4c97498bd301d84096da251c98a07c7723e65 Create write support
0d52aaab4479697da7686c15f77a3d64d9165190 One more thing
```

```
6d52a271eda8725415634dd79daabbc4d9b6008e Merge branch 'experiment'
0b7434d86859cc7b8c3d5e1dddfed66ff742fcbc Add commit function
4682c3261057305bdd616e23b64b0857d832627b Add todo file
166ae0c4d3f420721acbb115cc33848dfcc2121a Create write support
9fceb02d0ae598e95dc970b74767f19372d61af8 Update rakefile
964f16d36dfccde844893cac5b347e7b3d44abbc Commit the todo
8a5cbc430f1a9c3d00faaeffd07798508422908a Update readme
```

Теперь предположим, что вы забыли отметить версию проекта v1.2, которая была там, где находится коммит «Update rakefile». Вы можете добавить тег и позже. Для отметки коммита укажите его контрольную сумму (или её часть) как параметр команды:

```
$ git tag -a v1.2 9fceb02
```

Проверим, что коммит отмечен:

```
$ git tag
v0.1
v1.2
v1.3
v1.4
v1.4-lw
v1.5

$ git show v1.2
tag v1.2
Tagger: Scott Chacon <schacon@gee-mail.com>
Date:   Mon Feb 9 15:32:16 2009 -0800

version 1.2
commit 9fceb02d0ae598e95dc970b74767f19372d61af8
Author: Magnus Chacon <mchacon@gee-mail.com>
Date:   Sun Apr 27 20:43:35 2008 -0700

    Update rakefile
...
```

????? ??????

По умолчанию, команда `git push` не отправляет теги на удалённые сервера. После создания теги нужно отправлять явно на удалённый сервер. Процесс аналогичен отправке веток — достаточно выполнить команду `git push origin <tagname>`.

```
$ git push origin v1.5
Counting objects: 14, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (12/12), done.
Writing objects: 100% (14/14), 2.05 KiB | 0 bytes/s, done.
Total 14 (delta 3), reused 0 (delta 0)
To git@github.com:schacon/simplegit.git
 * [new tag]         v1.5 -> v1.5
```

Если у вас много тегов, и вам хотелось бы отправить все за один раз, то можно использовать опцию `--tags` для команды `git push`. В таком случае все ваши теги отправятся на удалённый сервер (если только их уже там нет).

```
$ git push origin --tags
Counting objects: 1, done.
Writing objects: 100% (1/1), 160 bytes | 0 bytes/s, done.
Total 1 (delta 0), reused 0 (delta 0)
To git@github.com:schacon/simplegit.git
 * [new tag]         v1.4 -> v1.4
 * [new tag]         v1.4-lw -> v1.4-lw
```

Теперь, если кто-то клонирует (clone) или выполнит `git pull` из вашего репозитория, то он получит вдобавок к остальному и ваши метки.

> `git push` отправляет оба типа тегов. Отправка тегов командой `git push <remote> --tags` не различает аннотированные и легковесные теги. В настоящее время не существует опции чтобы отправить только легковесные теги, но если использовать команду `git push <remote> --follow-tags`, то отправятся только аннотированные теги.

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

Для удаления тега в локальном репозитории достаточно выполнить команду `git tag -d <tagname>`. Например, удалить созданный ранее легковесный тег можно следующим образом:

```
$ git tag -d v1.4-lw
Deleted tag 'v1.4-lw' (was e7d5add)
```

Обратите внимание, что при удалении тега не происходит его удаления с внешних серверов. Существует два способа изъятия тега из внешнего репозитория.

Первый способ — это выполнить команду `git push <remote> :refs/tags/<tagname>`:

```
$ git push origin :refs/tags/v1.4-lw
To /git@github.com:schacon/simplegit.git
 - [deleted]         v1.4-lw
```

Это следует понимать как обновление внешнего тега пустым значением, что приводит к его удалению.

Второй способ убрать тег из внешнего репозитория более интуитивный:

```
$ git push origin --delete <tagname>
```

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

Если вы хотите получить версии файлов, на которые указывает тег, то вы можете сделать `git checkout` для тега. Однако, это переведёт репозиторий в состояние «detached HEAD», которое имеет ряд неприятных побочных эффектов.

```
$ git checkout v2.0.0
Note: switching to 'v2.0.0'.

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 performing another checkout.

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 99ada87... Merge pull request #89 from schacon/appendix-final
```

```
$ git checkout v2.0-beta-0.1
Previous HEAD position was 99ada87... Merge pull request #89 from schacon/appendix-final
HEAD is now at df3f601... Add atlas.json and cover image
```

Если в состоянии «detached HEAD» внести изменения и сделать коммит, то тег не изменится, при этом новый коммит не будет относиться ни к какой из веток, а доступ к нему можно будет получить только по его хешу. Поэтому, если вам нужно внести изменения — исправить ошибку в одной из старых версий — скорее всего вам следует создать ветку:

```
$ git checkout -b version2 v2.0.0
Switched to a new branch 'version2'
```

Если сделать коммит в ветке `version2`, то она сдвинется вперед и будет отличаться от тега `v2.0.0`, так что будьте с этим осторожны.

Revision #2
Created 2023-10-20 02:58:08 UTC by Антон Сергеевич Абраменко
Updated 2023-10-20 13:18:33 UTC by Антон Сергеевич Абраменко