

06 标签管理

发布一个版本时，我们通常先在版本库中打一个标签（**tag**），这样，就唯一确定了打标签时刻的版本。将来无论什么时候，取某个标签的版本，就是把那个打标签的时刻的历史版本取出来。所以，标签也是版本库的一个快照。

Git的标签虽然是版本库的快照，但其实它就是指向某个**commit**的指针（跟分支很像对不对？但是分支可以移动，标签不能移动），所以，创建和删除标签都是瞬间完成的。

Git有**commit**，为什么还要引入**tag**？

“请把上周一的那个版本打包发布，**commit**号是6a5819e...”

“一串乱七八糟的数字不好找！”

如果换一个办法：

“请把上周一的那个版本打包发布，版本号是v1.2”

“好的，按照**tag v1.2**查找**commit**就行！”

所以，**tag**就是一个让人容易记住的有意义的名字，它跟某个**commit**绑在一起。

创建标签

在Git中打标签非常简单，首先，切换到需要打标签的分支上：

```
$ git branch
* dev
  master
$ git checkout master
Switched to branch 'master'
```

然后，敲命令 **git tag** 就可以打一个新标签：

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

可以用命令 **git tag** 查看所有标签：

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

默认标签是打在最新提交的**commit**上的。有时候，如果忘了打标签，比如，现在已经是周五了，但应该在周一打的标签没有打，怎么办？

方法是找到历史提交的commit id，然后打上就可以了：

```
$ git log --pretty=oneline --abbrev-commit
12a631b (HEAD -> master, tag: v1.0, origin/master) merged
bug fix 101
4c805e2 fix bug 101
e1e9c68 merge with no-ff
f52c633 add merge
cf810e4 conflict fixed
5dc6824 & simple
14096d0 AND simple
b17d20e branch test
d46f35e remove test.txt
b84166e add test.txt
519219b git tracks changes
e43a48b understand how stage works
1094adb append GPL
e475afc add distributed
eaadf4e wrote a readme file
```

比方说要对 `add merge` 这次提交打标签，它对应的commit id是 `f52c633`，敲入命令：

```
$ git tag v0.9 f52c633
```

再用命令 `git tag` 查看标签：

```
$ git tag
v0.9
v1.0
```

注意，标签不是按时间顺序列出，而是按字母排序的。可以用 `git show` 查看标签信息：

```
$ git show v0.9
commit f52c63349bc3c1593499807e5c8e972b82c8f286 (tag:
v0.9)
Author: Michael Liao <askxuefeng@gmail.com>
Date:   Fri May 18 21:56:54 2018 +0800

    add merge

diff --git a/readme.txt b/readme.txt
...
```

可以看到，`v0.9` 确实打在 `add merge` 这次提交上。

还可以创建带有说明的标签，用 `-a` 指定标签名，`-m` 指定说明文字：

```
$ git tag -a v0.1 -m "version 0.1 released" 1094adb
```

用命令 `git show` 可以看到说明文字：

```
$ git show v0.1
tag v0.1
Tagger: Michael Liao <askxuefeng@gmail.com>
Date:   Fri May 18 22:48:43 2018 +0800

version 0.1 released

commit 1094adb7b9b3807259d8cb349e7df1d4d6477073 (tag:
v0.1)
Author: Michael Liao <askxuefeng@gmail.com>
Date:   Fri May 18 21:06:15 2018 +0800

    append GPL

diff --git a/readme.txt b/readme.txt
...
```

注意：标签总是和某个commit挂钩。如果这个commit既出现在master分支，又出现在dev分支，那么在这两个分支上都可以看到这个标签。

小结

- 命令 `git tag` 用于新建一个标签，默认为 `HEAD`，也可以指定一个 commit id；
- 命令 `git tag -a -m "blablabla..."` 可以指定标签信息；
- 命令 `git tag` 可以查看所有标签。

操作标签

如果标签打错了，也可以删除：

```
$ git tag -d v0.1
Deleted tag 'v0.1' (was f15b0dd)
```

因为创建的标签都只存储在本地，不会自动推送到远程。所以，打错的标签可以在本地安全删除。

如果要推送某个标签到远程，使用命令 `git push origin`：

```
$ git push origin v1.0
Total 0 (delta 0), reused 0 (delta 0)
To github.com:michaelliao/learngit.git
 * [new tag]          v1.0 -> v1.0
```

或者，一次性推送全部尚未推送到远程的本地标签：

```
$ git push origin --tags
Total 0 (delta 0), reused 0 (delta 0)
To github.com:michaelliao/learngit.git
* [new tag]          v0.9 -> v0.9
```

如果标签已经推送到远程，要删除远程标签就麻烦一点，先从本地删除：

```
$ git tag -d v0.9
Deleted tag 'v0.9' (was f52c633)
```

然后，从远程删除。删除命令也是push，但是格式如下：

```
$ git push origin :refs/tags/v0.9
To github.com:michaelliao/learngit.git
- [deleted]          v0.9
```

要看看是否真的从远程库删除了标签，可以登陆GitHub查看。

小结

- 命令 `git push origin` 可以推送一个本地标签；
- 命令 `git push origin --tags` 可以推送全部未推送过的本地标签；
- 命令 `git tag -d` 可以删除一个本地标签；
- 命令 `git push origin :refs/tags/` 可以删除一个远程标签。