Git Tagging

Create signed, unsigned or lightweight tags to permanantly mark important points in your project history.

Like most VCSs, Git has the ability to 'tag' specific points in history as being important - generally people use this to mark release points ('v1.0', etc). In this lesson we will learn how to list the available tags, how to create new tags, and what the different types of tags in Git are.

listing tags

Simply listing the available tags in Git is very straightforward. Just type 'git tag'.

```
$ git tag
v0.1
v1.3
```

This will just list them in alphabetical order, so there is no real importance in what order they list out in.

You can also search for tags with a particular pattern. In the Git source repo itself, for instance, there are over 240 tags. If you're only interested in looking at the 1.4.2 series, you could run this:

```
$ ./git tag -l v1.4.2.*
v1.4.2.1
v1.4.2.2
v1.4.2.3
v1.4.2.4
```

creating tags

There are a two main types of tags in Git - lightweight and annotated. Lightweight tags are very much like branches that don't change - it's just a pointer to a specific commit. Annotated tags, however, are stored as full objects in the Git database. They are checksummed, contain the tagger name, email and date, have a tagging message and can be GPG signed and verified. It's generally recommended to create annotated tags so you can have all this information, but if you want a temporary tag or for some reason don't want to keep that other information, lightweight tags are available too.

annotated tags

Creating an annotated tag in Git is very simple. The easiest way is just to specify a **-a** when you run the 'tag' command.

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

The ‘-m’ specifies a tagging message, which is stored with the tag. If you don’t specify it for an annotated tag, Git will launch your editor so you can type it in.

You can see the tag data along with the commit that was tagged by using the ‘git show’ command:

```
$ git show v1.4
tag v1.4
Tagger: Scott Chacon <schacon@gmail.com>
Date:   Mon Feb 9 14:45:11 2009 -0800

my version 1.4
commit 15027957951b64cf874c3557a0f3547bd83b3ff6
Merge: 4a447f7... a6b4c97...
Author: Scott Chacon <schacon@gmail.com>
Date:   Sun Feb 8 19:02:46 2009 -0800

    Merge branch 'experiment'
```

That shows the tagger information, date it was tagged and the annotation message before showing the commit information.

signed tags

You can also sign your tags with GPG (http://www.gnupg.org/faq.html), assuming you have a private key and everything. All you have to do is use **-s** instead of -a.

```
[master]$ git tag -s v1.5 -m 'my signed 1.5 tag'

You need a passphrase to unlock the secret key for
user: "Scott Chacon <schacon@gmail.com>"
1024-bit DSA key, ID F721C45A, created 2009-02-09
```

Then if you run a ‘git show’ on that tag, you can see your GPG signature attached to it.

```
[master]$ git show v1.5
tag v1.5
Tagger: Scott Chacon <schacon@gmail.com>
Date:   Mon Feb 9 15:22:20 2009 -0800

my signed 1.5 tag
-----BEGIN PGP SIGNATURE-----
Version: GnuPG v1.4.8 (Darwin)
```

```
iEYEABECAAYFAkmQurIACgkQON3DxfchxFr5cACeIMN+ZxLKggJQf0QYiQBwgySN
Ki0An2JeAVUCAiJ7Ox6ZEtK+NvZAj82/
=WryJ
-----END PGP SIGNATURE-----
commit 15027957951b64cf874c3557a0f3547bd83b3ff6
Merge: 4a447f7... a6b4c97...
Author: Scott Chacon <schacon@gmail.com>
Date:   Sun Feb 8 19:02:46 2009 -0800

    Merge branch 'experiment'
```

A bit later, we'll learn how to verify signed tags.

lightweight tags

Another way to tag commits is with a 'lightweight' tag. This is basically just the commit checksum stored in a file - no other information is kept. To create a lightweight tag, simply don't supply the '-a', '-s' or '-m' options.

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

This time if we run 'git show' on that tag, we won't see the extra tag information, it will just show the commit.

```
$ git show v1.4-lw
commit 15027957951b64cf874c3557a0f3547bd83b3ff6
Merge: 4a447f7... a6b4c97...
Author: Scott Chacon <schacon@gmail.com>
Date:   Sun Feb 8 19:02:46 2009 -0800

    Merge branch 'experiment'
```

verifying tags

To verify a signed tag, you just use 'git tag -v (tag)'. That will use gpg to verify the signature. You'll need the signers public key in your keyring.

```
$ git tag -v v1.4.2.1
object 883653babd8ee7ea23e6a5c392bb739348b1eb61
type commit
tag v1.4.2.1
tagger Junio C Hamano <junkio@cox.net> 1158138501 -0700
```

```
GIT 1.4.2.1

Minor fixes since 1.4.2, including git-mv and git-http with alternates.
gpg: Signature made Wed Sep 13 02:08:25 2006 PDT using DSA key ID F3119B9A
gpg: Good signature from "Junio C Hamano <junkio@cox.net>"
gpg:                 aka "[jpeg image of size 1513]"
Primary key fingerprint: 3565 2A26 2040 E066 C9A7  4A7D C0C6 D9A4 F311 9B9A
```

If you don't have the signers public key, you'll get something like this instead:

```
gpg: Signature made Wed Sep 13 02:08:25 2006 PDT using DSA key ID F3119B9A
gpg: Can't check signature: public key not found
error: could not verify the tag 'v1.4.2.1'
```

tagging later

You can also tag commits after you've moved past them. If my commit history looks like this

```
$ git log --pretty=oneline
15027957951b64cf874c3557a0f3547bd83b3ff6 Merge branch 'experiment'
a6b4c97498bd301d84096da251c98a07c7723e65 beginning write support
0d52aaab4479697da7686c15f77a3d64d9165190 one more thing
6d52a271eda8725415634dd79daabbc4d9b6008e Merge branch 'experiment'
0b7434d86859cc7b8c3d5e1dddfed66ff742fcbc added a commit function
4682c3261057305bdd616e23b64b0857d832627b added a todo file
166ae0c4d3f420721acbb115cc33848dfcc2121a started write support
9fceb02d0ae598e95dc970b74767f19372d61af8 updated rakefile
964f16d36dfccde844893cac5b347e7b3d44abbc commit the todo
8a5cbc430f1a9c3d00faaeffd07798508422908a updated readme
```

And I forgot to tag the project at 'v1.2', which was at the 'updated rakefile' commit, I can add it after the fact. To tag that commit, you can just specify the commit checksum (or part of it) at the end of the command.

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

Now we can see that we've tagged the commit:

```
$ 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@gmail.com>
```

```
Date:   Mon Feb 9 15:32:16 2009 -0800

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

    updated rakefile
...
```

sharing tags

By default, the ‘git push’ command will not transfer tags to remote servers. To do so, you have to explicitly add a **–tags** to the ‘git push’ command.

```
[master]$ git push --tags
Counting objects: 50, done.
Compressing objects: 100% (38/38), done.
Writing objects: 100% (44/44), 4.56 KiB, done.
Total 44 (delta 18), reused 8 (delta 1)
To git@github.com:schacon/simplegit.git
 * [new tag]         v0.1 -> v0.1
 * [new tag]         v1.2 -> v1.2
 * [new tag]         v1.4 -> v1.4
 * [new tag]         v1.4-lw -> v1.4-lw
 * [new tag]         v1.5 -> v1.5
```

And now when someone else clones or pulls from your repository, they will get all your tags as well.

« previous (branching.html)

next » (log.html)