Git Tagging

Like most of the Version Control Systems, Git also has the ability to tag a specific points in a repository's history. Most of the people use this for marking various version or release points (v1.0,v1.1 and so on) of the system.

Listing of tags

Listing the existing tags in git is straightforward. Just type *`git tag`* (with optional -l or –list):

```
$ git tag
v1.0
v1.2
```

You can also search for tags that match a particular pattern. The Git source repo, for instance, contains more than 500 tags. If you're interested only in looking at the 1.8.5 series, you can run this:

```
$ git tag -l "v1.8.5"
v1.8.5
v1.8.5-rc.0
v1.8.5-rc.1
```

Creating tags

Git supports two types of tags: *lightweight* and *annotated*.

A lightweight tag is very much like a branch that doesn't change — it's just a pointer to a specific commit.

Annotated tags, however, are stored as full objects in the Git database. They're checksummed; contain the tagger name, email, and date; have a tagging message; and can be signed and verified with GNU Privacy Guard (GPG). It's generally recommended that you 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 the other information, lightweight tags are available too.

Annotated tags

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

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

The -m specifies a tagging message, which is stored with the tag. If you don't specify a message for an annotated tag, Git launches 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: 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
```

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

Lightweight tags
Another way to tag commits is with a lightweight tag. This is basically the commit checksum stored in a file — no other information is kept. To create a lightweight tag, don’t supply any of the -a, -s, or -m options, just provide a tag name:

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

This time, if you run`*git show*`on the tag, you don’t see the extra tag information. The command just shows the commit:

```
$ 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
```

Deleting tags

To delete a tag on your local repository, you can use git tag -d <tagname>. For example, we could remove our lightweight tag above as follows:

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

Note that this does not remove the tag from any remote servers. There are two common variations for deleting a tag from a remote server.

The first variation is 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
```

The way to interpret the above is to read it as the null value before the colon is being pushed to the remote tag name, effectively deleting it.

The second (and more intuitive) way to delete a remote tag is with:

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