**Git Tags**

Tagging lets developers mark important checkpoints in the course of their projects' development. For instance, software release versions can be tagged. (Ex: v1.3.2) It essentially allows you to give a commit a special name(tag).

To view all the created tags in alphabetical order:

git tag

To get more information on a tag:

git show v1.4

There are two types of tags:

Annotated

git tag -a v1.2 -m "my version 1.4"

Lightweight

git tag v1.2

They differ in the way that they are stored.

cdcThese create tags on your current commit.

Incase, you’d like to tag a previous commit specify the commit ID you’d like to tag:

git tag -a v1.2 9fceb02

The tags names may be used instead of commit IDs while checking out and pushing commits to a remote repo.

You can list all available tags in a project with the git tag command (nate that they will appear in alphabetical order):

$ git tag

v1.0

v2.0

v3.0

This way of listing tags is great for small projects, but greater projects can have hundreds of tags, so you may need to filter them when searching for an important point in the history. You can find tags containing specific characters adding an -l to the git tag command:

$ git tag -l "v2.0\*"

v2.0.1

v2.0.2

v2.0.3

v2.0.4

**Create a tag**

You can create two type of tags: annotated and lightweight. They first ones are compete objects in GIT database: they are checksummed, requiere a message (like commits) and store other important data such as name, email and date. On the other hand, lightweight tags dont require a mesage or store other data, working just as a pointer to a specific point in the project.

**Create an annotated tag**

To create an anotated tag, add -a tagname -m "tag message" to the git tag command:

$ git tag -a v4.0 -m "release version 4.0"

$ git tag

v1.0

v2.0

v3.0

v4.0

As you can see, the -a specifies that you are creating an annotated tag, after comes the tag name and finally, the -m followed by the tag message to store in the Git database.

**Create a lightweight tag**

Lightweight tags contain only the commit checksum (no other information is stored). To create one, just run the git tag command without any other options (the -lw characters at the end of the name are used to indicate lightweight tags, but you can mark them as you like):

$ git tag v4.1-lw

$ git tag

v1.0

v2.0

v3.0

v4.0

v4.1-lw

This time you didn’t specify a message or other relevant data, so the tag contains only the refered commit’s checksum.

**View tag’s data**

You can run the git show command to view the data stored in a tag. In the case of annotated tags, you’ll see the tag data and the commit data:

$ git show v4.0

tag v4.0

Tagger: John Cash <john@cash.com>

Date: Mon Sat 28 15:00:25 2017 -0700

release version 4.0

commit da43a5fss745av88d47839247990022a98419093

Author: John Cash <john@cash.com>

Date: Fri Feb 20 20:30:05 2015 -0700

finished details

If the tag you are watching is a lightweight tag, you’ll only see the refered commit data:

$ git show v1.4-lw

commit da43a5f7389adcb9201ab0a289c389ed022a910b

Author: John Cash <john@cash.com>

Date: Fri Feb 20 20:30:05 2015 -0700

finished details

**Tagging old commits**

You can also tag past commits using the git tag commit. In order to do this, you’ll need to specify the commit’s checksum (or at least a part of it) in the command’s line.

First, run git log to find out the required commit’s checksum:

$ git log --pretty=oneline

ac2998acf289102dba00823821bee04276aad9ca added products section

d09034bdea0097726fd8383c0393faa0072829a7 refactorization

a029ac120245ab012bed1ca771349eb9cca01c0b modified styles

da43a5f7389adcb9201ab0a289c389ed022a910b finished details

0adb03ca013901c1e02174924486a08cea9293a2 small fix in search textarea styles

When you have the checksum needed, add it at the end of the tag creation line:

$ git tag -a v3.5 a029ac

You’ll see the tag was correctly added running git tag:

$ git tag

v1.0

v2.0

v3.0

v3.5

v4.0

v4.1-lw

**Push tags**

Git does’t push tags by default when you run the git push command. So, to succesfully push a tag to a server you’ll have to git push origin command:

$ git push origin v4.0

Counting objects: 14, done.

Delta compression using up to 8 threads.

Compressing objects: 100% (16/16), done.

Writing objects: 100% (18/18), 3.15 KiB | 0 bytes/s, done.

Total 18 (delta 4), reused 0 (delta 0)

To git@github.com:jcash/gitmanual.git

\* [new tag] v4.0 -> v4.0

You can also use the --tags option to add multiple tags at once with the git push origin command:

$ 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:jcash/gitmanual.git

\* [new tag] v4.0 -> v4.0

\* [new tag] v4.1-lw -> v4.1-lw

**Checking out Tags**

You can use git checkout to checkout to a tag like you would normally do. But you need to keep in mind that this would result a *detached HEAD* state.

$ git checkout v0.0.3

Note: checking out 'v0.0.3'.

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.

**Deleting a Tag**

You may find a situation were you want to delete a certain tag. There’s a very useful command for this situations:

$ git tag --delete v0.0.2

$ git tag

v0.0.1

v0.0.3

v0.0.4