Identifiers and Tags

coursera.org/learn/git-distributed-development/supplement/JkFCx/identifiers-and-tags

Every time you do a commit, git assigns a unique 160-bit 40-character hexadecimal hash value to it. While you can refer to those commits with these values, it is obviously unwieldy. For instance, taking an example from the Linux kernel source repository, we can see the history of commits with:

```
$ git log | grep "^commit" | head -10

commit 56b24d1bbcff213dc9e1625eea5b8e13bb50feb8

commit 5a45a5a881aeb82ce31dd1886afe04146965df23

commit ecade114250555720aa8a6232b09a2ce2e47ea99

commit 2c4ea6e28dbf15ab93632c5c189f3948366b8885

commit 106e4da60209b508894956b6adf4688f84c1766d

commit 4b050f22b5c68fab3f96641249a364ebfe354493

commit 84c37c168c0e49a412d7021cda3183a72adac0d0

commit 0acf611997d9d05dbfb559c3c6e379c861eb5957

commit 434fd6353b4c83938029ca6ea7dfa4fc82d602bd

commit 85298808617299fe713ed3e03114058883ce3d8a
```

If you want to refer to or revert to a certain commit, typing in such a long string is obviously a pain. You can instead create and use a tag. Thus, you could do:

```
$ git tag ver_10 08d869aa8683703c4a60fdc574dd0809f9b073cd
```

or, even better:

```
$ git tag ver_10 08d869
```

If you have used a long enough part of the 40-character string in the second example, so that the reference is unique, then the short version will suffice.

git tag creates a tag or annotated tag (a text string that references a commit object). The tag is placed in **.git/refs/tags** unless it is an annotated tag, in which case, the tag is created as an object in the object store (changes induced by **git tag**):

Command	Source Files	Index	Commit Chain	References
git tag	Unchanged	Unchanged	Unchanged	A new tag is created

We will talk about checking things out later, but all you would have to do to revert to the development point labeled by **ver_10** would be:

```
$ git checkout ver_10
```