

## Tagging

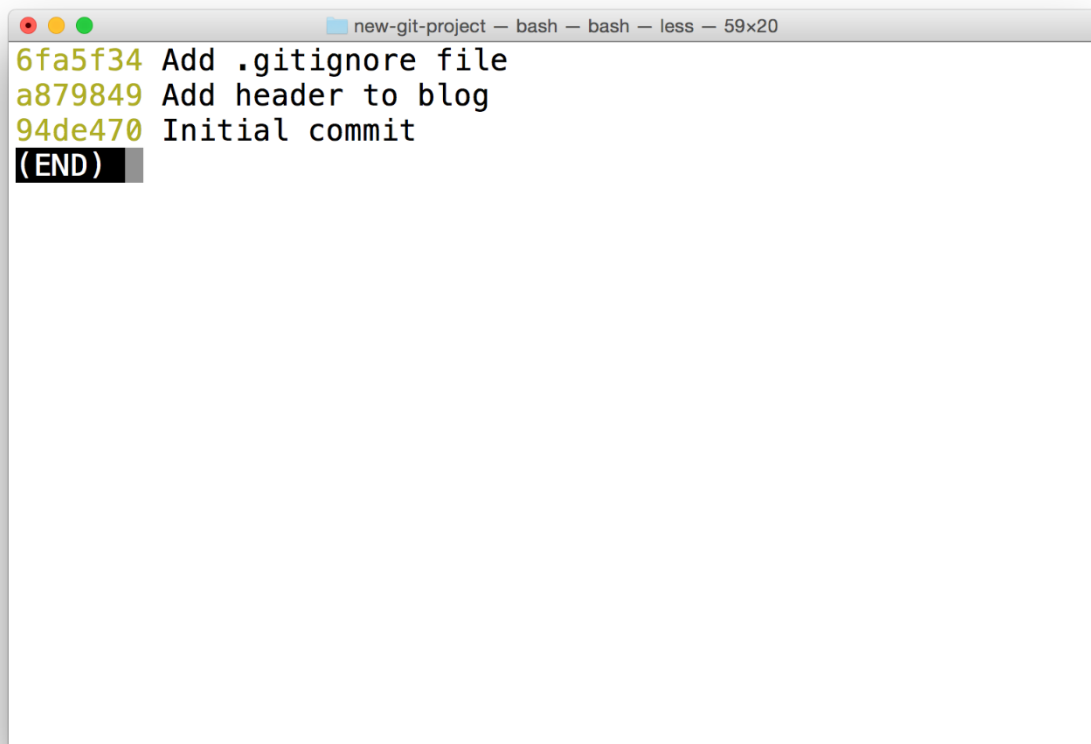
So far in this course, we've been zoomed in on the specific Git commands. We've learned how they work in detail and what it looks like running them on the Terminal.

Let's zoom out a bit to look at how a Git tag fits into a repository.

### Where Are We?

You can do these steps in either project, but I'll be doing them in the new-git-project project.

Let's take a look at the log of the project so far:

A screenshot of a macOS Terminal window. The title bar at the top reads "new-git-project — bash — bash — less — 59x20". The terminal content shows the output of a git log command in oneline format. It lists three commits: a yellow SHA hash "6fa5f34" followed by the message "Add .gitignore file", a yellow SHA hash "a879849" followed by "Add header to blog", and a yellow SHA hash "94de470" followed by "Initial commit". Below these, the text "(END)" is displayed in white on a black background, indicating the end of the log output.

```
new-git-project — bash — bash — less — 59x20
6fa5f34 Add .gitignore file
a879849 Add header to blog
94de470 Initial commit
(END)
```

*The Terminal application showing the output from running `git log --oneline`.*

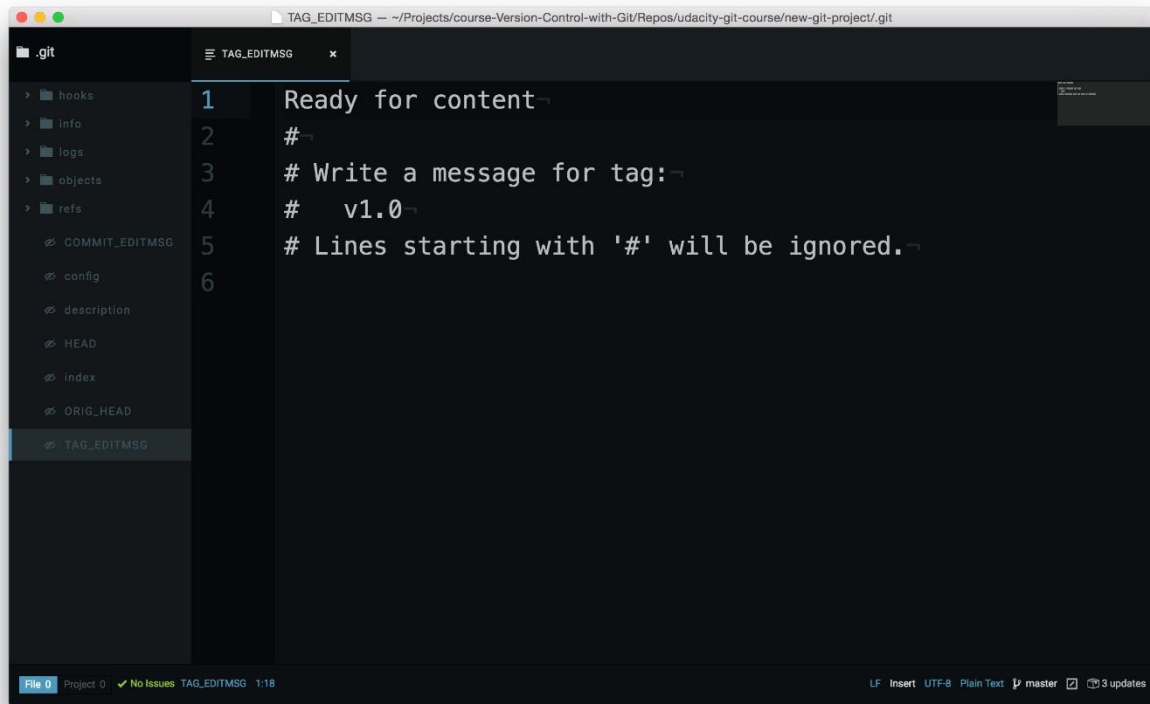
### Git Tag Command

Pay attention to what's shown (just the SHA and the commit message)

The command we'll be using to interact with the repository's tags is the `git tag` command:

```
$ git tag -a v1.0
```

This will open your code editor and wait for you to supply a message for the tag. How about the message "Ready for content"?



*Code editor waiting for the tag's message to be supplied.*

CAREFUL: In the command above (`git tag -a v1.0`) the `-a` flag is used. This flag tells Git to create an *annotated* tag. If you don't provide the flag (i.e. `git tag v1.0`) then it'll create what's called a *lightweight* tag.

Annotated tags are recommended because they include a lot of extra information such as:

- the person who made the tag
- the date the tag was made
- a message for the tag

Because of this, you should always use annotated tags.

## Verify Tag

After saving and quitting the editor, nothing is displayed on the command line. So how do we know that a tag was actually added to the project? If you type out just `git tag`, it will display all tags that are in the repository.

```
richardkalehoff (master) new-git-project
$ git tag
v1.0
richardkalehoff (master) new-git-project
$
```

*The Terminal application showing the output of the git tag command. The tag v1.0 is listed.*

So we've verified that it's in the repository, but let's actually see *where* it is inside the repository. To do that, we'll go back to our good old friend, git log!

### Git Log's --decorate Flag

As you've learned, git log is a pretty powerful tool for letting us check out a repository's commits. We've already looked at a couple of its flags, but it's time to add a new one to our toolbelt. The --decorate flag will show us some details that are hidden from the default view.

Try running git log --decorate now!

### 💡 --decorate Flag Changes in Git 2.13 💡

In the 2.13 update to Git, the log command has changed to automatically enable the --decorate flag. This means that you do not need to include the --decorate flag in your command, since it is automatically included, anyway! So the following commands result in the exact same output:

```
$ git log --decorate
```

```
$ git log
```

Check out [the 2.13 release notes](#).

```
new-git-project — bash — less — 77x20
commit 6fa5f34790808d9f4dccc0fa8fdbbc40760102d6e (HEAD -> master, tag: v1.0)
Author: Richard Kalehoff <richardkalehoff@gmail.com>
Date:   Wed Dec 21 14:15:14 2016 -0500

    Add .gitignore file

commit a8798493c8507fc24146fcf2e5837c703ddce08a
Author: Richard Kalehoff <richardkalehoff@gmail.com>
Date:   Mon Dec 19 09:27:33 2016 -0500

    Add header to blog

commit 94de47077674e39d454167a97c198f7c60a72de4
Author: Richard Kalehoff <richardkalehoff@gmail.com>
Date:   Fri Dec 16 15:55:52 2016 -0500

    Initial commit

~
~
(END)
```

*The Terminal application showing the output of the `git log --decorate` command. The log output now displays the newly created tag.*

The tag information is at the very end of the first line:

```
commit 6fa5f34790808d9f4dccc0fa8fdbbc40760102d6e (HEAD -> master, tag: v1.0)
```

See how it says tag: v1.0? That's the tag! Remember that tags are associated with a specific commit. This is why the tag is on the same line as the commit's SHA.

### HEAD -> master?

Did you notice that, in addition to the tag information being displayed in the log, the `--decorate` also revealed HEAD -> master? That's information about a *branch*! We'll be looking at branches in Git, next.

### Deleting A Tag

What if you accidentally misspelled something in the tag's message, or mistyped the actual tag name (v0.1 instead of v1.0). How could you fix this? The easiest way is just to delete the tag and make a new one.

A Git tag can be deleted with the `-d` flag (for *delete*!) and the name of the tag:

```
$ git tag -d v1.0
```

```
richardkalehoff (master) new-git-project
$ git tag
v1.0
richardkalehoff (master) new-git-project
$ git tag -d v1.0
Deleted tag 'v1.0' (was 76631c3)
richardkalehoff (master) new-git-project
$
```

*The Terminal application showing the removal of a tag by using the -d flag. The command that is run is `git tag -d v1.0`.*

## Adding A Tag To A Past Commit

Running `git tag -a v1.0` will tag the most recent commit. But what if you wanted to tag a commit that occurred farther back in the repo's history?

All you have to do is provide the SHA of the commit you want to tag!

```
$ git tag -a v1.0 a87984
```

(after popping open a code editor to let you supply the tag's message) this command will tag the commit with the SHA a87084 with the tag v1.0. Using this technique, you can tag any commit in the entire git repository! Pretty neat, right?...and it's just a simple addition to add the SHA of a commit to the Git tagging command you already know.

## Git Tag Recap

To recap, the git tag command is used to add a marker on a specific commit. The tag does not move around as new commits are added.

```
$ git tag -a beta
```

This command will:

- add a tag to the most recent commit
- add a tag to a specific commit *if a SHA is passed*

## Further Research

- [Git Basics - Tagging](#) from the Git Book
- [Git Tag](#) from the Git Docs