

Git Tag

Git has the ability to tag specific points in history as being important. Typically people use this functionality to mark release points (v1.0, and so on). In this section, you'll learn how to list the available tags, how to create new tags, and what the different types of tags are.

Creating Tags:

Git supports two types of tags: 1. Lightweight 2. Annotated.

Annotated : 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.

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

```
$ git tag -a v1.0 -m "my release version 1.0"
```

```
[ec2-user@ip-172-31-27-131 JMSTechhome]$ git tag -a v1.0 -m "my release version 1.0"  
[ec2-user@ip-172-31-27-131 JMSTechhome]$ git tag  
v1.0
```

Check the list of tags use

```
$ git log
```

you can see the tag data along with the commit that was tagged by using the **git show** command:

```
$ git show v1.0
```

```
[ec2-user@ip-172-31-27-131 JMSTechhome]$ git show v1.0  
tag v1.0  
Tagger: ybmadhu <ybmadhu707@gmail.com>  
Date: Tue Dec 11 10:16:52 2018 +0000  
  
my release version 1.0  
  
commit 5eb85a2daff9736d868ae78e3f988d81b0dd0095 (HEAD -> master, tag: v1.0)  
Author: EC2 Default User <ec2-user@ip-172-31-27-131.ap-south-1.compute.internal>  
Date: Tue Dec 11 10:13:32 2018 +0000  
  
latest file added  
  
diff --git a/latest.txt b/latest.txt  
new file mode 100644  
index 0000000..64f3359  
--- /dev/null  
+++ b/latest.txt  
@@ -0,0 +1,3 @@  
+fdjsfhf  
+fkdsfj  
+
```

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

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 v1.1
```

```
[ec2-user@ip-172-31-27-131 JMSTechhome]$ git tag v1.1
[ec2-user@ip-172-31-27-131 JMSTechhome]$ git tag
v1.0
v1.1
```

Tagging Later:

Take hash code using git log

```
git tag v0.0.1 3f42f2d
```

You can also tag commits after you've moved past them. Suppose your commit history looks like this:

```
$ git log --pretty=oneline
```

```
[ec2-user@ip-172-31-27-131 JMSTechhome]$ git log --pretty=oneline
5eb85a2daff9736d868ae78e3f988d81b0dd0095 (HEAD -> master, tag: v1.1, tag: v1.0) latest file added
0f66581c5e51b289c2c9a82aa32744baef0544f8 (origin/master, origin/HEAD) this is new file2.txt
c1c5b5c22dfa1e48b7f462ed6c6fe4b673ebf2f8 added file
fe3217ffa94828f0ebb25baa930334a191917cc6 Initial Commit
[ec2-user@ip-172-31-27-131 JMSTechhome]$ git tag -a v0.1 fe3217
[ec2-user@ip-172-31-27-131 JMSTechhome]$ git tag
v0.1
v1.0
v1.1
```

Sharing Tags

By default, the git push command doesn't transfer tags to remote servers. You will have to explicitly push tags to a shared server after you have created them. This process is just like sharing remote branches — you can run like below.

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

```
$ git push origin v1.0
```

```
[ec2-user@ip-172-31-27-131 JMSTechhome]$ git push origin v1.0
Username for 'https://github.com': ybmadhu
Password for 'https://ybmadhu@github.com':
Counting objects: 4, done.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (4/4), 512 bytes | 512.00 KiB/s, done.
Total 4 (delta 0), reused 0 (delta 0)
To https://github.com/ybmadhu/JMSTechhome.git
 * [new tag]          v1.0 -> v1.0
```

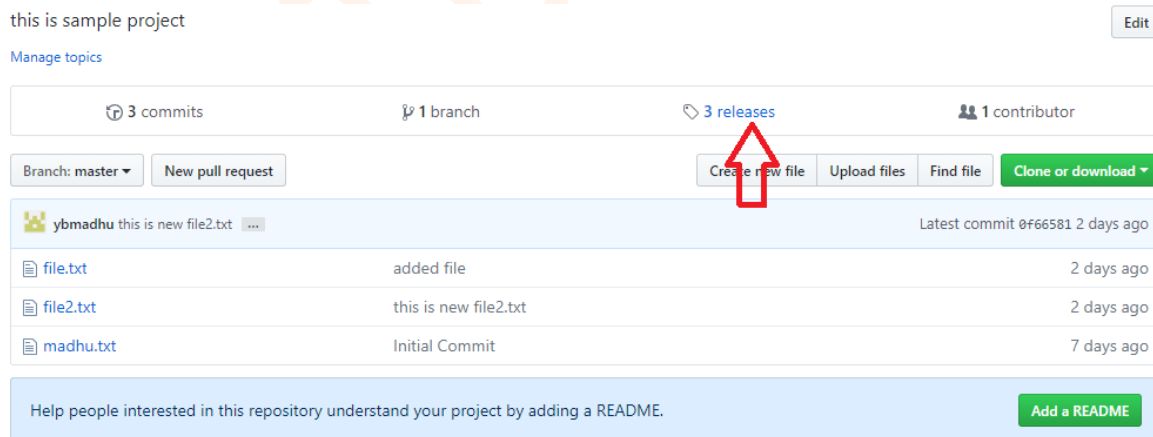
If you have a lot of tags that you want to push up at once, you can also use the **--tags** option to the **git push** command. This will transfer all of your tags to the remote servers that are not already there.

```
$ git push origin --tags
```

```
[ec2-user@ip-172-31-27-131 JMSTechhome]$ git push origin --tags
Username for 'https://github.com': ybmadhu
Password for 'https://ybmadhu@github.com':
Counting objects: 1, done.
Writing objects: 100% (1/1), 162 bytes | 162.00 KiB/s, done.
Total 1 (delta 0), reused 0 (delta 0)
To https://github.com/ybmadhu/JMSTechhome.git
 * [new tag]          v0.1 -> v0.1
 * [new tag]          v1.1 -> v1.1
```

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

Check in git hub our tags are pushed or not.



this is sample project [Edit](#)

[Manage topics](#)

3 commits 1 branch 3 releases 1 contributor

Branch: master [New pull request](#) [Create new file](#) [Upload files](#) [Find file](#) [Clone or download](#)

ybmaddhu this is new file2.txt	Latest commit 0f66581 2 days ago
file.txt	added file 2 days ago
file2.txt	this is new file2.txt 2 days ago
madhu.txt	Initial Commit 7 days ago

Help people interested in this repository understand your project by adding a README. [Add a README](#)

Click the release link it will open a tag list in github.

Releases Tags

Draft a new release

2 hours ago v1.1
5eb85a2 zip tar.gz

2 hours ago v1.0
5eb85a2 zip tar.gz

2 hours ago v0.1
fe3217f zip tar.gz

© 2018 GitHub, Inc. Terms Privacy Security Status Help

Contact GitHub Pricing API Training Blog About

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 v0.1
```

```
[ec2-user@ip-172-31-27-131 JMSTechhome]$ git tag -d v0.1  
Deleted tag 'v0.1' (was 4204bac)
```

Note that this does not remove the tag from any remote servers. In order to update any remotes, you must use **git push <remote> :refs/tags/<tagname>**

```
$ git push origin :refs/tags/v1.0
```

```
[ec2-user@ip-172-31-27-131 JMSTechhome]$ git push origin :refs/tags/v1.0  
Username for 'https://github.com': ybmadhu  
Password for 'https://ybmadhu@github.com':  
To https://github.com:ybmadhu/JMSTechhome.git  
- [deleted] v1.0
```

Delete all tags at time:

- #Delete local tags.
- `git tag -d $(git tag -l)`
- #Fetch remote tags.
- `git fetch`
- #Delete remote tags.

- x # Pushing once should be faster than multiple times
- #Delete local tags.
- `git tag -d $(git tag -l)`

Remote deletes

- `git ls-remote --tags origin | awk '/^(.*) (\s+) (.*[a-z0-9])$/ {print ":" $2}' | xargs git push origin`