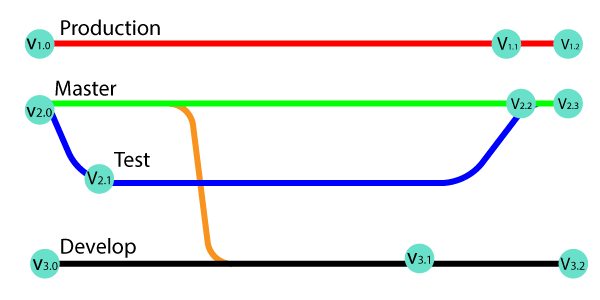
Git Tags

Tags make a point as a specific point in Git history. Tags are used to mark a commit stage as relevant. We can tag a commit for future reference. Primarily, it is used to mark a project's initial point like v1.1.

Tags are much like branches, and they do not change once initiated. We can have any number of tags on a branch or different branches. The below figure demonstrates the tags on various branches.



In the above image, there are many versions of a branch. All these versions are tags in the repository.

There are two types of tags.

* Annotated tag
* Light-weighted tag

Both of these tags are similar, but they are different in case of the amount of Metadata stores.

When to create a Tag:

* When you want to create a release point for a stable version of your code.
* When you want to create a historical point that you can refer to reuse in the future.

Git Create tag

To create a tag first, checkout to the branch where you want to create a tag. To check out the branch, run the below command:

1. $ git checkout **<Branch** name**>**

Now, you are on your desired branch, say, master. Consider the below output:

Git Tags

You can create a tag by using the git tag command. Create a tag with some name say **v1.0, v1.1,** or any other name whatever you want. To create a tag, run the command as follows:

**Syntax:**

1. $ git tag **<tag** name**>**

The above command will mark the current status of the project. Consider the below example:

1. $ git tag projectv1.0

Git Tags

The above command will create a mark point on the master branch as **projectv1.0**.

Git List Tag

We can list the available tags in our repository. There are three options that are available to list the tags in the repository. They are as follows:

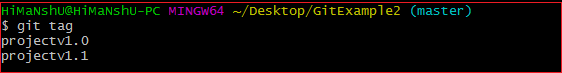
* git tag
* git show
* git tag -l ".\*"

The "git tag":

It is the most generally used option to list all the available tags from the repository. It is used as:

1. $ git tag

**Output:**



As we can see from the above output, the git tag command is listing the available tags from the repository.

The git tag show <tagname>:

It's a specific command used to display the details of a particular tag. It is used as:

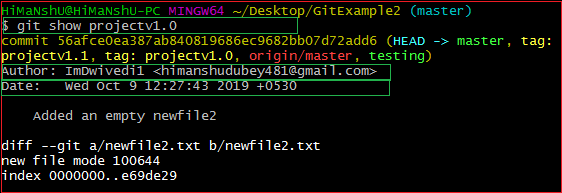
**Syntax:**

1. $ git tag show **<tagname>**

The above command will display the tag description, consider the below command:

1. $ git tag show projectv1.0

**Output:**



In the above output, the git show tag is displaying the description of tag **projectv1.0**, such as author name and date.

The git tag -l ".\*":

It is also a specific command-line tool. It displays the available tags using wild card pattern. Suppose we have ten tags as v1.0, v1.1, v1.2 up to v1.10. Then, we can list all v pattern using tag pattern v. it is used as:

**Syntax:**

1. $ git tag -l "**<pattern>**.\*"

The above command will display all the tags that contain wild card characters. Consider the below command:

1. $ git tag -l "pro\*"

**Output:**



The above command is displaying a list of the tags that started with a word **pro**.

Types of Git tags

There are two types of tags in git. They are as:

* Annotated tag
* Light-weighted tag

Let's understand both of these tags in detail.

Annotated Tags

Annotated tags are tags that store extra Metadata like developer name, email, date, and more. They are stored as a bundle of objects in the Git database.

If you are pointing and saving a final version of any project, then it is recommended to create an annotated tag. But if you want to make a temporary mark point or don't want to share information, then you can create a light-weight tag. The data provided in annotated tags are essential for a public release of the project. There are more options available to annotate, like you can add a message for annotation of the project.

To create an annotated tag, run the below command:

**Syntax:**

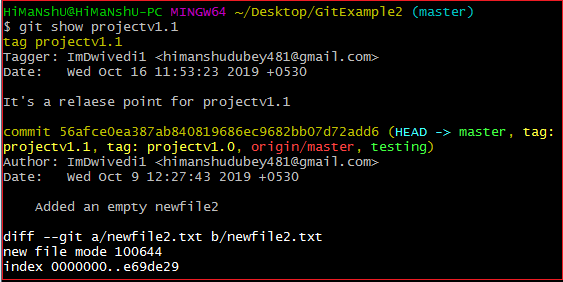
1. $ git tag **<tag** name**>** -m "**<** **Tag** message**>**

The above command will create a tag with a message. Annotated tags contain some additional information like author name and other project related information. Consider the below image:

Git Tags

The above command will create an annotated tag **projectv1.1** in the master branch of my project's repository.

When we display an annotated tag, it will show more information about tag. Consider the below output:



Light-Weighted Tag:

Git supports one more type of tag; it is called as Light-weighted tag. The motive of both tags is the same as marking a point in the repository. Usually, it is a commit stored in a file. It does not store unnecessary information to keep it light-weight. No command-line option such as **-a,-s** or **-m** are supplied in light-weighted tag, pass a tag name.

**Syntax:**

1. $ git tag **<tag** name**>**

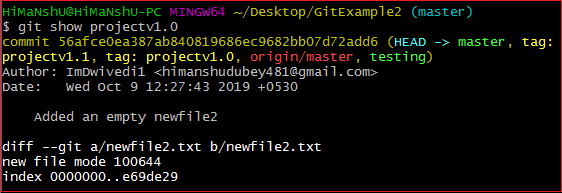
The above command will create a light-weight tag. Consider the below example:

1. $ git tag projectv1.0

Git Tags

The given output will create a light-weight tag named **projectv1.0**.

It will display a reduced output than an annotated tag. Consider the below output:



Git Push Tag

We can push tags to a remote server project. It will help other team members to know where to pick an update. It will show as **release point** on a remote server account. The git push command facilitates with some specific options to push tags. They are as follows:

* Git push origin <tagname>
* Git push origin -tags/ Git push --tags

The git push origin :

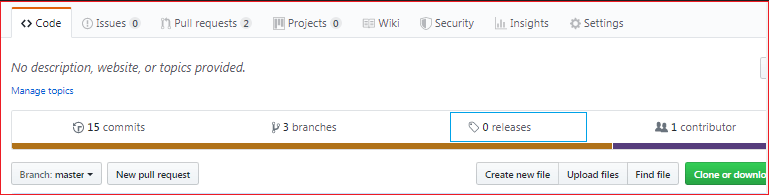
We can push any particular tag by using the git push command. It is used as follows:

**Syntax:**

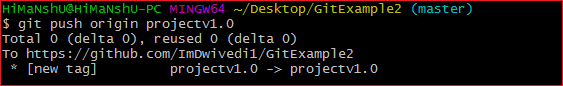
1. $ git push origin **<tagname>**

The above command will push the specified tag name as a release point. Consider the below example:

I have created some tags in my local repository, and I want to push it on my GitHub account. Then, I have to operate the above command. Consider the below image; it is my remote repository current status.



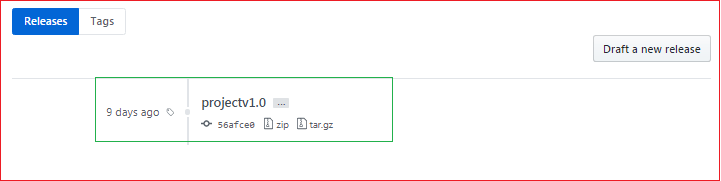
The above image is showing the release point as **0 releases**. Now, perform the above command. Consider the below output:



I have pushed my projectv1.0 tag to the remote repository. It will change the repository's current status. Consider the below image:



By refreshing the repository, it is showing release point as **1 release**. We can see this release by clicking on it. It will show as:



We can download it as a zip and tar file.

The git push origin --tag/ git push --tags:

The given command will push all the available tags at once. It will create as much release point as the number of tags available in the repository. It is used as follows:

**Syntax:**

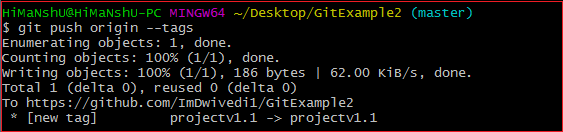
1. $ git push origin --tags

Or

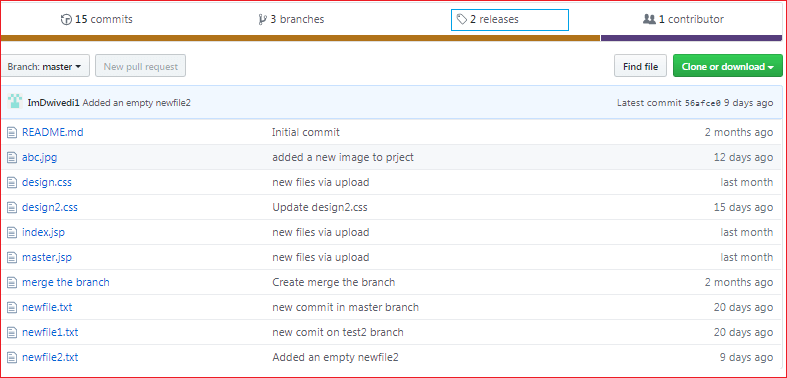
1. $ git push --tags

The above command will push all the available tags from the local repository to the remote repository. Consider the below output:

**Output:**



Tags have been pushed to remote server origin; thus, the release point is also updated. Consider the below snapshot of the repository:



The release point is updated in the above output according to tags. You can see that releases updated as **2 releases**.

Git Delete Tag

Git allows deleting a tag from the repository at any moment. To delete a tag, run the below command:

**Syntax:**

1. $git tag --d **<tagname>**

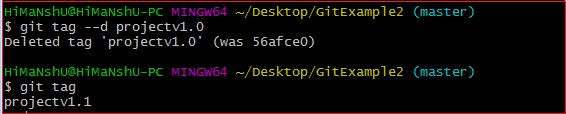
Or

1. $ git tag --delete **<tagname>**

The above command will delete a particular tag from the local repository. Suppose I want to delete my tag **projectv1.0** then the process will be as:

1. $ git tag --d projectv1.0

Consider below output:



The tag projectv1.0 has been deleted from the repository.

Delete a Remote Tag:

We can also delete a tag from the remote server. To delete a tag from the remote server, run the below command:

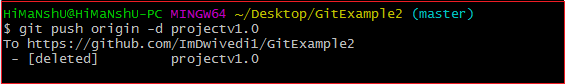
**Syntax:**

1. $ git push origin -d **<tagname>**

Or

1. $ git push origin --delete**<tag** name**>**

The above command will delete the specified tag from the remote server. Consider the below output:



The **projectv1.0** tag has been deleted from the remote server origin.

Delete Multiple Tags:

We can delete more than one tag just from a single command. To delete more than one tag simultaneously, run the below command:

**Syntax:**

1. $ git tag -d **<tag1>** **<tag2>**

**Output:**

The above command will delete both the tags from the local repository.

We can also delete multiple tags from the remote server. To delete tags from the server origin, run the below command:

1. $ git push origin -d **<tag1>** **<tag2>**

The above command will delete both tags from the server.

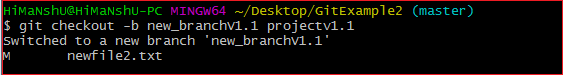
Git Checkout Tags

There is no actual concept of check out the tags in git. However, we can do it by creating a new branch from a tag. To check out a tag, run the below command:

**Syntax:**

1. $ git checkout -b **<** **new** branch name**>** **<tag** name**>**

The above command will create a new branch with the same state of the repository as it is in the tag. Consider the below output:



The above command will create a new branch and transfer the status of the repository to **new\_branchv1.1** as it is on tag projectv1.1.

**Create a tag from an older commit:**

If you want to go back to your history and want to create a tag on that point. Git allows you to do so. To create a tag from an older commit, run the below command:

1. **<** **git** tag **<tagname>** **<** **reference** of commit**>**

In the above command, it is not required to give all of the 40 digit number; you can give a part of it.

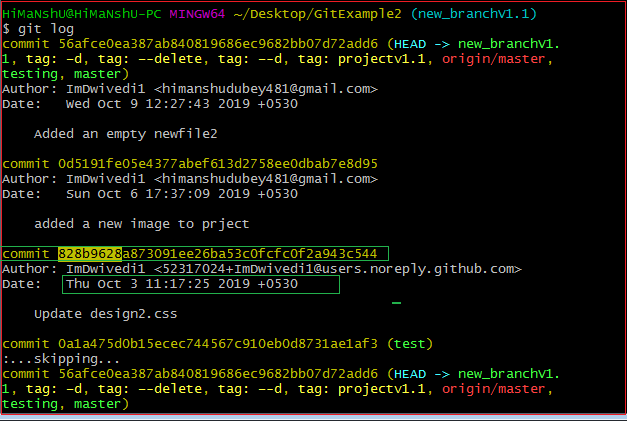
Suppose I want to create a tag for my older commit, then the process will be as follows:

**Check the older commits:**

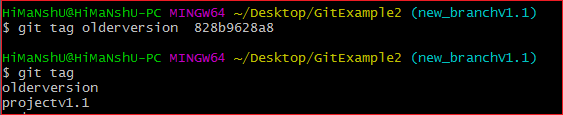
To check the older commit, run the git status command. It will operate as follows:

1. $ git status

Consider the below output:



The above output is showing the older commits. Suppose I want to create a tag for my commit, starting with **828b9628**. Copy the particular reference of the commit. And pass it as an argument in the above command. Consider the below output:



In the above output, an earlier version of the repository is tagged as an **olderversion**.