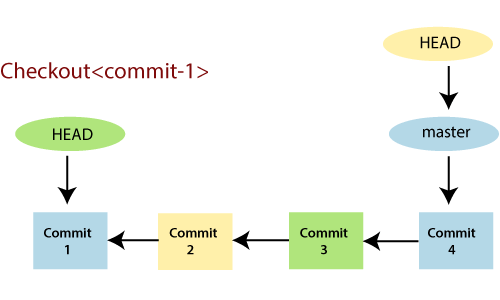
# Git Head

The **HEAD** points out the last commit in the current checkout branch. It is like a pointer to any reference. The HEAD can be understood as the "**current branch**." When you switch branches with 'checkout,' the HEAD is transferred to the new branch.



The above fig shows the HEAD referencing commit-1 because of a 'checkout' was done at commit-1. When you make a new commit, it shifts to the newer commit. The git head command is used to view the status of Head with different arguments. It stores the status of Head in **.git\refs\heads** directory. Let's see the below example:

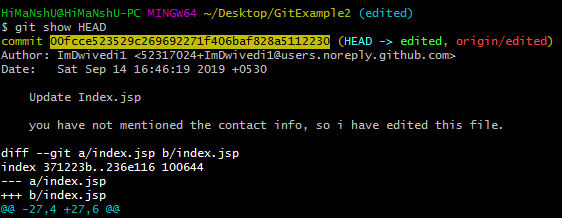
## Git Show Head

The **git show head** is used to check the status of the Head. This command will show the location of the Head.

**Syntax:**

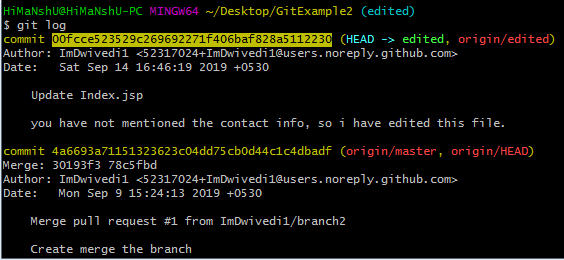
1. $ git show HEAD

**Output:**



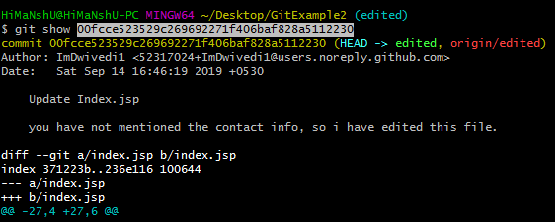
In the above output, you can see that the commit id for the Head is given. It means the Head is on the given commit.

Now, check the commit history of the project. You can use the git log command to check the commit history. See the below output:



As we can see in the above output, the commit id for most recent commit and Head is the same. So, it is clear that the last commit has the Head.

We can also check the status of the Head by the commit id. Copy the commit id from the above output and paste it with the **git show** command. Its result is same as **git show head** command if the commit id is last commit's id. See the below output:



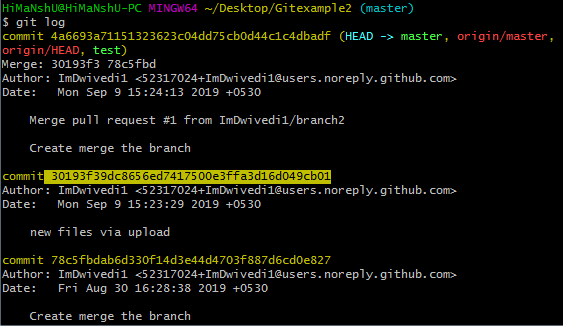
The above output is the same as git show output.

The HEAD is capable of referring to a specific revision that is not associated with a branch name. This situation is called a detached HEAD.

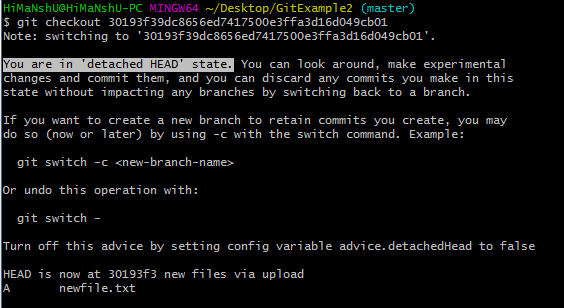
## Git Detached Head

GitHub keeps track of all commits or snapshots over time. If you check the 'git log' in your terminal, you can show all the previous commits up to the first commit. Detached HEAD mode allows you to discover an older state of a repository. It is a natural state in Git.

When **Head doesn't point to most recent commit, such state is called detached Head**. If you checkout with an older commit, it will stand the detached head condition. See the below example:



I have copied the older commit id. Now I will check out with this id.



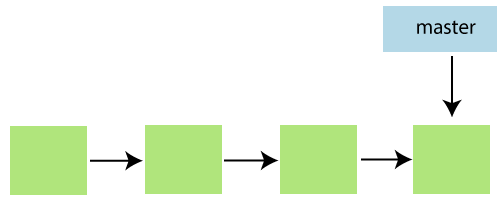
As you can see in the given example, Head does not point the most recent commit. It is called a detached head state. It is always recommended, do not commit on detached Head.

# Git Origin Master

The term "git origin master" is used in the context of a remote repository. It is used to deal with the remote repository. The term origin comes from where repository original situated and master stands for the main branch. Let's understand both of these terms in detail.

## Git Master

Master is a naming convention for Git branch. It's a default branch of Git. After cloning a project from a remote server, the resulting local repository contains only a single local branch. This branch is called a "master" branch. It means that "master" is a repository's "default" branch.

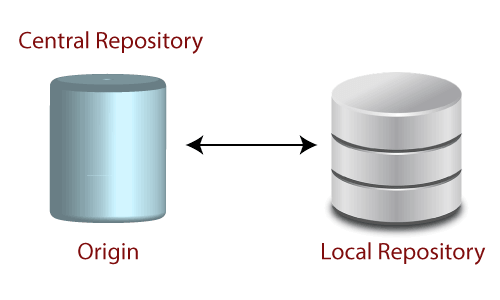


In most cases, the master is referred to as the main branch. Master branch is considered as the final view of the repo. Your local repository has its master branch that always up to date with the master of a remote repository.

Do not mess with the master. If you edited the master branch of a group project, your changes will affect everyone else and very quickly there will be merge conflicts.

## Git Origin

In Git, The term origin is referred to the remote repository where you want to publish your commits. The default remote repository is called **origin**, although you can work with several remotes having a different name at the same time. It is said as an alias of the system.



The origin is a short name for the remote repository that a project was initially being cloned. It is used in place of the original repository URL. Thus, it makes referencing much easier.

Origin is just a standard convention. Although it is significant to leave this convention untouched, you could ideally rename it without losing any functionality.

In the following example, the URL parameter acts as an origin to the "clone" command for the cloned local repository:

1. $ git clone https://github.com/ImDwivedi1/Git-Example

Some commands in which the term origin and master are widely used are as follows:

* Git push origin master
* Git pull origin master

Git has two types of branches called local and remote. To use git pull and git push, you have to tell your local branch that on which branch is going to operate. So, the term origin master is used to deal with a remote repository and master branch. The term **push origin master** is used to push the changes to the remote repository. The term **pull origin master** is used to access the repository from remote to local.