Git Rm

In Git, the term rm stands for remove. It is used to remove individual files or a collection of files. The key function of git rm is to remove tracked files from the Git index. Additionally, it can be used to remove files from both the working directory and staging index.

The files being removed must be ideal for the branch to remove. No updates to their contents can be staged in the index. Otherwise, the removing process can be complex, and sometimes it will not happen. But it can be done forcefully by **-f** option.

Let's understand it with an example.

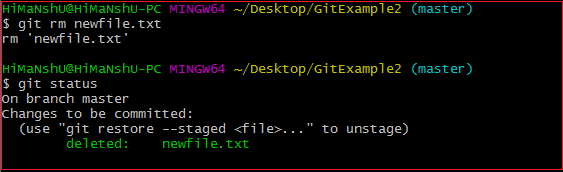
The git rm command

The git rm command is used to remove the files from the working tree and the index.

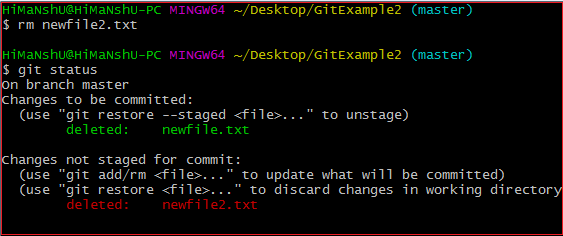
If we want to remove the file from our repository. Then it can be done by the git rm command. Let's take a file say newfile.txt to test the rm command. The git rm command will be operated as:

1. $ git rm <file Name>

The above command will remove the file from the Git and repository. The git rm command removes the file not only from the repository but also from the staging area. If we check the status of the repository, then it will show as deleted. Consider the below output:



In the above output, the file **newfile.txt** has been removed from the version control system. So the repository and the status are shown as deleted. If we use only **the rm command**, then it will not permanently delete the file from the Git. It can be tracked in the staging area. Consider the below output:



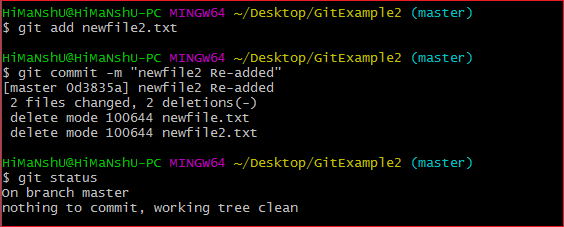
In the above output, the file newfile2.txt has been deleted. But when we check the status of the repository, we can track the file in the staging area. It means the newfile2 yet not deleted from the staging area, and it is also available in the repository. We can get it back on the version control system by committing it. To commit the file, first, add it to the index and then commit it. To add this file in the index, run the below command:

1. $ git add newfile2.txt

The above command will add the file to the index. To commit it, run the below command:

1. $ git commit -m "commit message."

It will commit the file and make it available to the version control system. Consider the below output:



In the above output, we are retrieving the file from the staging area to our directory. The newfile2.txt is re-added to our repository.

Git Rm Cached

Sometimes you want to remove files from the Git but keep the files in your local repository. In other words, you do not want to share your file on Git. Git allows you to do so. The cached option is used in this case. It specifies that the removal operation will only act on the staging index, not on the repository. The git rm command with cached option will be uses as:

1. $ git rm --cached <file name>

The above command will remove a file from the version control system. The deleted file will remain in the repository. Somehow this command will act as rm command. Let's understand it with an example.

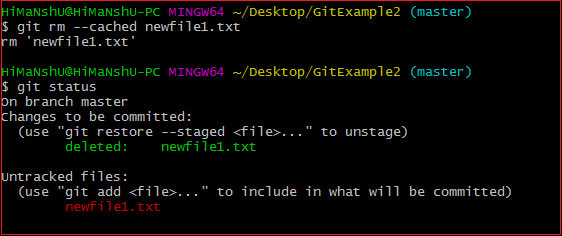
Suppose we want to remove a file from Git, take **newfile1.txt** for operation to remove this file, use the below command:

1. $ git rm --cached newfile1.txt

The above command will delete the file from the version control system, but still, it can be tracked in the repository. It also can be re-added on the version control system. To check the file status, use the status command as:

1. $ git status

Consider the below output:



As we can see from the above output, the newfile1.txt file is deleted from the version control system, but it can be tracked in the repository. This file is available on the version control system as an untracked file. We can track it by committing it.

Undo the Git Rm Command

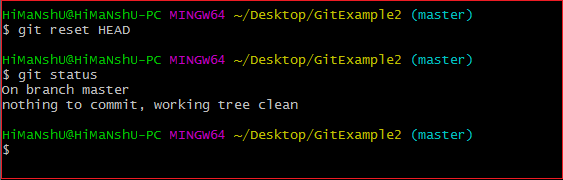
Execution of git rm command is not permanent; it can be reverted after execution. These changes cannot be persisted until a new commit is made on the repository. We can undo the git rm command. There are several ways to do so. The most usual and straight-forward way is git reset command. The git reset command will be used as follows:

1. $ git reset HEAD

Or we can also use:

1. $ git reset --hard

The above command will reset the position of the head. So that it will get the position of its just previous point. Consider the below output:



From the above output, we can see that the file has been successfully reset to its previous position.

There is another way to undo the git rm command. We can also do it by git checkout command. A checkout has the same effect and restores the latest version of a file from HEAD. It will be used as follows:

1. $ git checkout.

Limits of Git Rm command

The git rm is operated only on the current branch. The removing process is only applied to the working directory and staging index trees. It is not persisted in the repository history until a new commit is created.