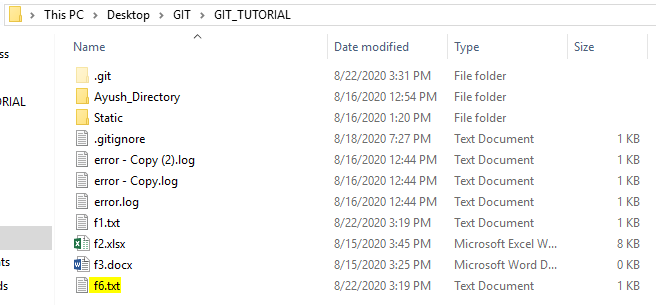
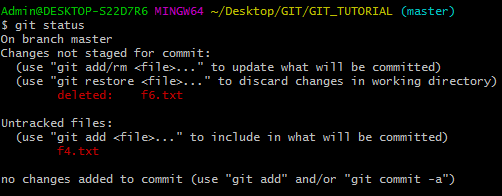
Suppose if we rename any file that is committed. And then checking the status of those:

* **Renaming manually**



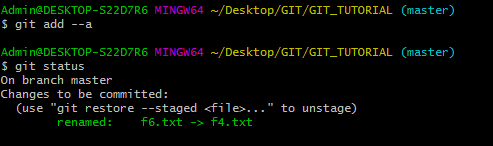
Renaming f6 to f4.

On rechecking the status:



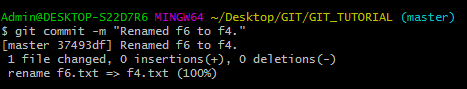
It is saying that file f6.txt has been deleted and f4.txt has been added as a new file. However, the file f6 has been renamed to f4. So GIT didn’t tracked this on checking the status.

On running add command and then checking status:



We can see that on reaching to staging area, GIT identified that f6 has been renamed to f4.

On performing commit:



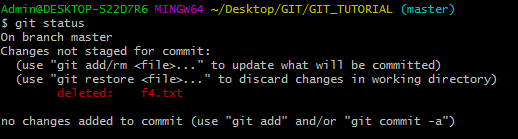
The f6 has been renamed to f4.

* **Removing a file using GIT**

We can remove the file directly from local machine or via GIT command as well.

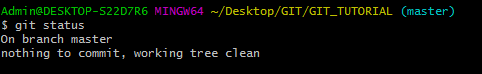
Let’s remove f4.txt.

As we removed f4 from machine and checked the GIT status, it shows that f4 has been removed.



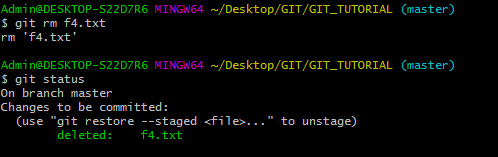
Let’s undo the changes and restore the file locally:

On doing so and checking the status:



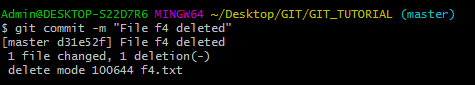
It states that everything is committed and clean.

We can remove the file directly using GIT command. Like this we will save effort to inform the change to the staging area (running add after removing from local manually).



The file had been removed from local machine and also the information had been provided to staging area.

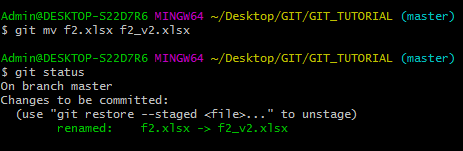
On committing the changes:



* **Renaming a file using GIT command**

Previously we renamed the file manually and then ran add command to notify staging area. This effort could be saved by executing **mv** command.

Let’s rename f2.xlsx:



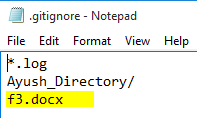
Here we directly renamed the file and that has also been staged. So mv command works as rename or move command. **We should always remember that GIT commands are inspired from UNIX.**

Making Tracked file as untracked and ignoring it:

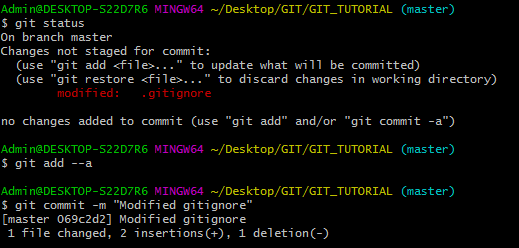
Suppose we have a file that we were tracking and it got committed. Now suppose we have done some changes to the file. But we think that we should ignore the changes and should not commit the changes.

Example:

File f3.docx is committed and we want to ignore it. We will mention the file name in .gitignore.



On checking and committing GIT status:

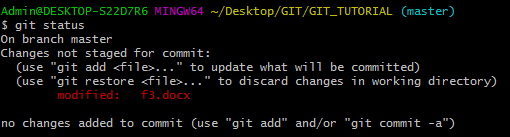


Now let’s modify the f3 file (it is added to ignore) and see whether it is getting tracked or not:



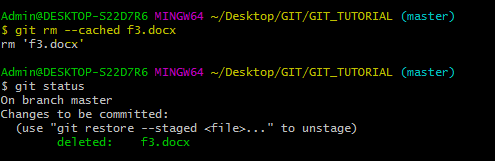


On checking status:

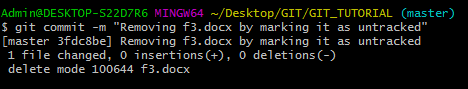


It is still tracking f3 file. We were in impression that this file would not be tracked as its name has been mentioned in git ignore.

This is because GIT has marked this file to track. It is not an untrack file. Hence we need to run a command to mark it untracked.

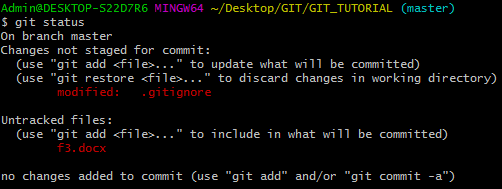


This will mark the file as untracked and now it would be ignored. We can see that in staging, it has been mentioned that f3.docx has been deleted.



**This file will not be tracked by GIT but it will be placed in local directory.**

If we remove the file name from git ignore, so again it would be shown as untracked.



We can see that git ignore is in modified block and f3.docx is now marked as Untracked.