**GIT BRANCH, MERGE AND REBASE**

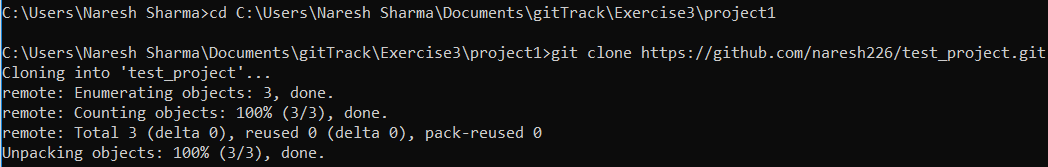
|  |  |
| --- | --- |
| **Action** | **Command Used** |
| Clone a git repository | Git clone <remote-git-repo-url> |
| Create branch | git checkout -b <branch-name> |
| Check status | git status |
| Add untracked files to staging area | git add . |
| Commit | git commit -m “message” |
| Add files to staging area and commit in one go | git commit -am “message” |
| Push all changes to remote | git push origin --all |
| Push changes of a branch to remote | git push origin <branch-name> |
| Switch branch | git checkout <branch-name> |
| Check difference | git diff <file-name> or git diff head |
| Merge branch | git merge <branch-name> |
| Rebase branch | git rebase <branch-name> |
| Pull changes from remote | git pull |
| Display all commits | git log |
| Create a track branch | git checkout --track origin/<branch-name> |

1. Create two folders project1 and project2 locally. Create a repository in github or bitbucket, name it test\_project.

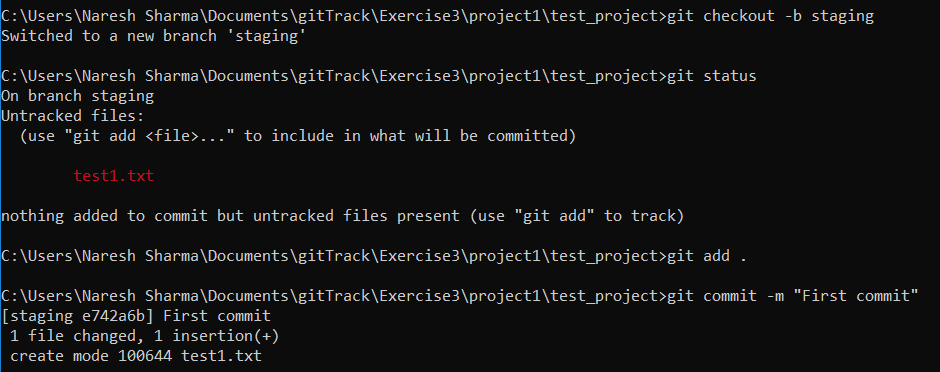
Created remote repository on github with name ‘test\_project’.

URL :- <https://github.com/naresh226/test_project>

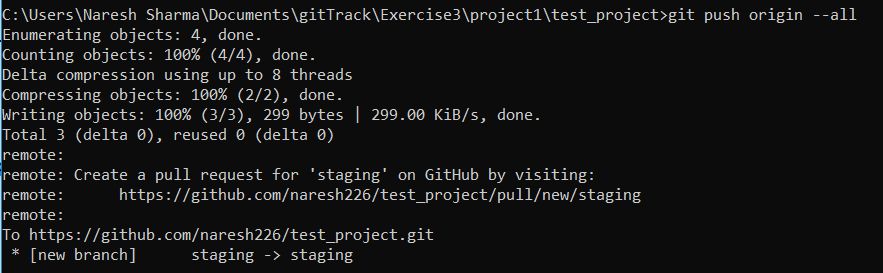
1. Switch to project1



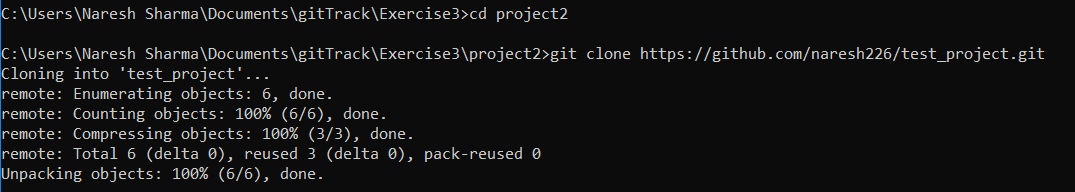
1. Switch to a new branch - 'staging'.
2. Add a file in it - 'test1' and add content to it 'This is first commit'.



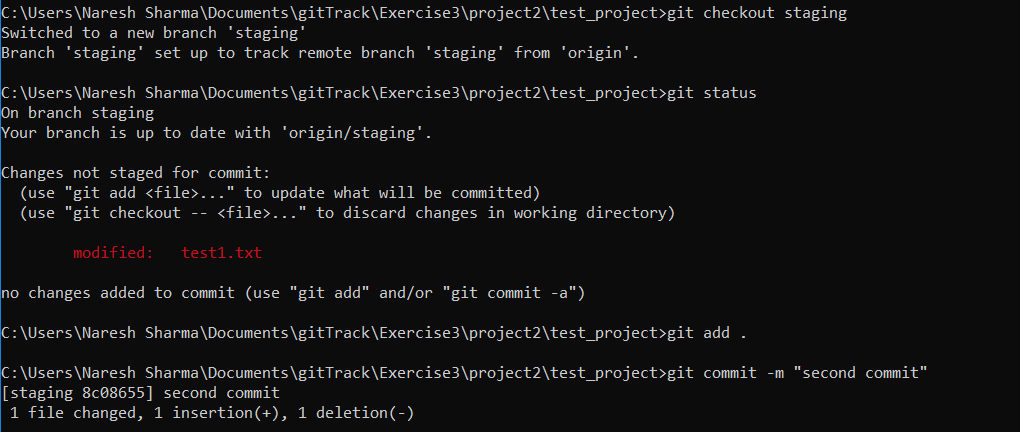
1. Push it to github.



1. Move to folder project2.
2. Clone the repo in project2.

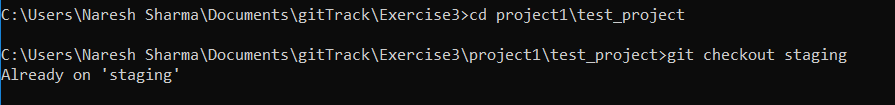


1. Switch to staging.
2. Change the text in file 'test1' to 'This is second commit'.
3. Commit and push to remote branch.

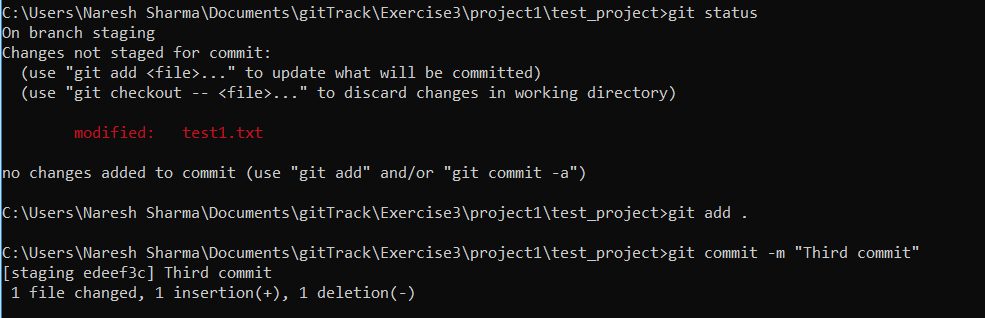




1. Move to project1.
2. Switch to staging branch.

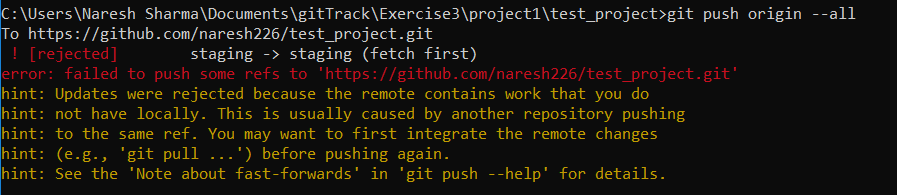


1. Change the text in file 'test1' to 'This is third commit'.

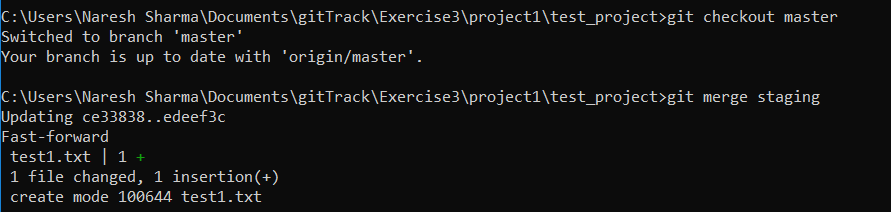


1. Commit and push to remote branch. (Explain what happened here).

There were changes pushed to the same reference by the other repository i.e from project2. So, to push the changes to remote first we need to pull the changes to current repository then push the combined changes.  
If we use force push (git push -f ) then, the current changes will be pushed overwriting the existing changes on remote.



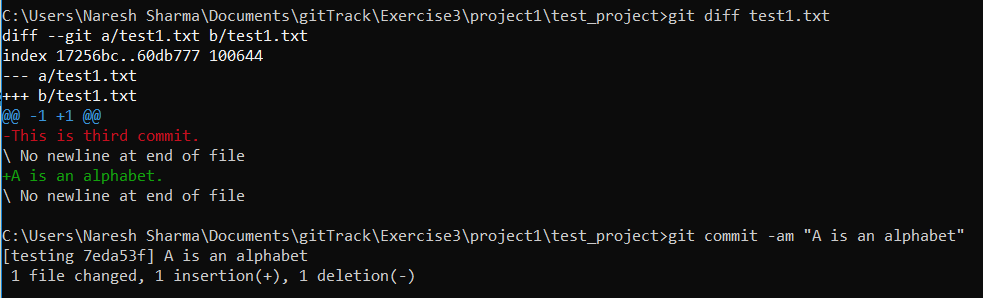
1. Switch to master.
2. Merge staging branch into master.



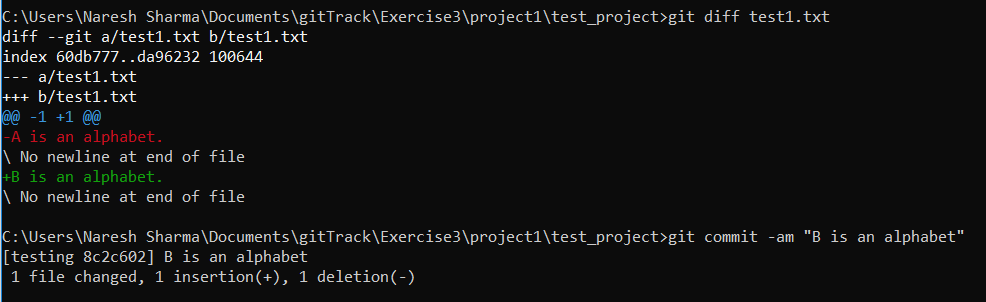
1. Create a new branch testing.

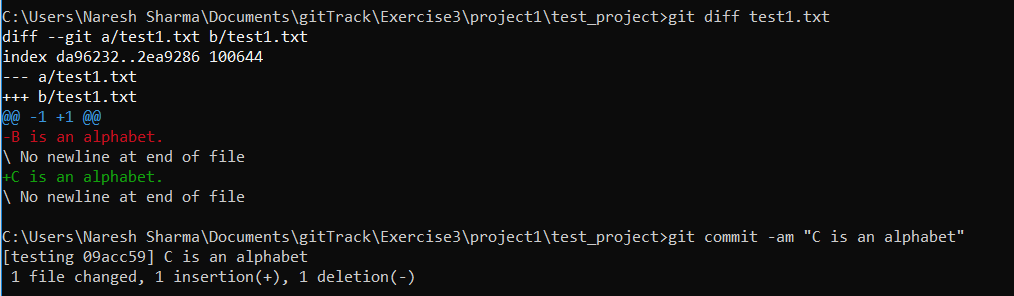


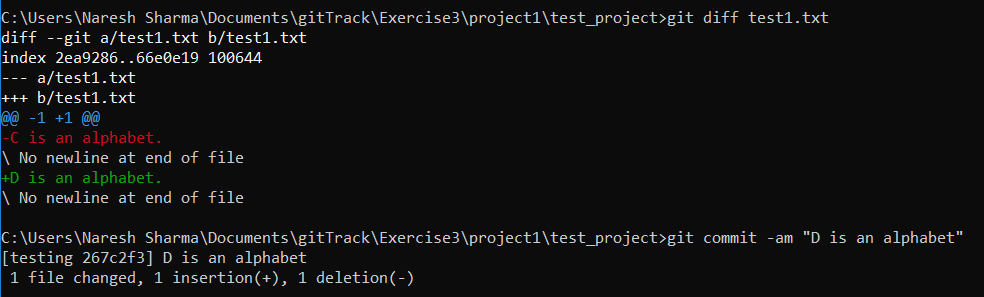
1. Change the text to 'A is an alphabet' and commit.

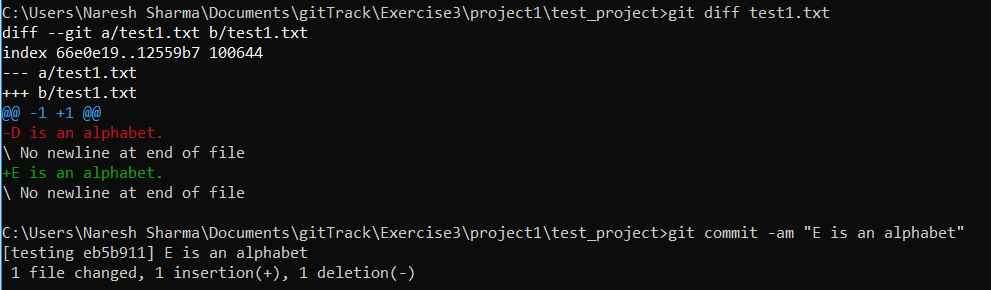


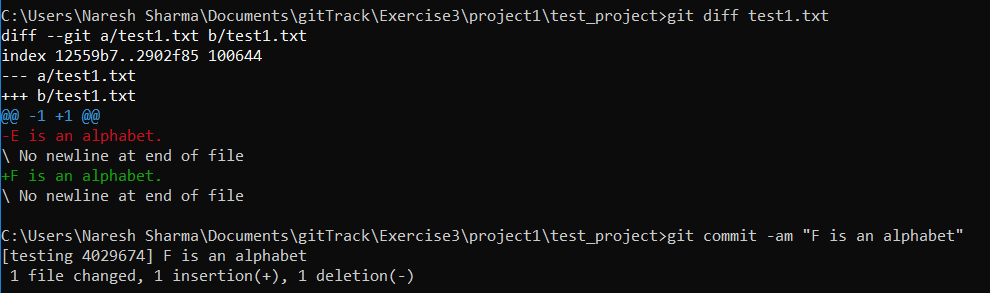
1. Repeat the 'step 18' seven times and each time update the alphabet 'A' to 'B' to 'C' etc.(Don't forget to commit the change everytime)

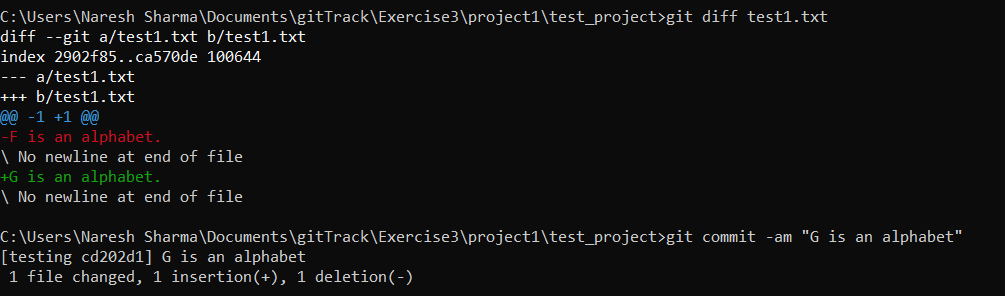


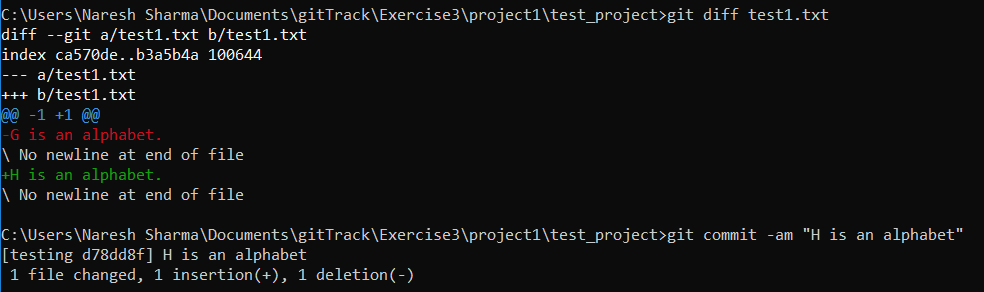




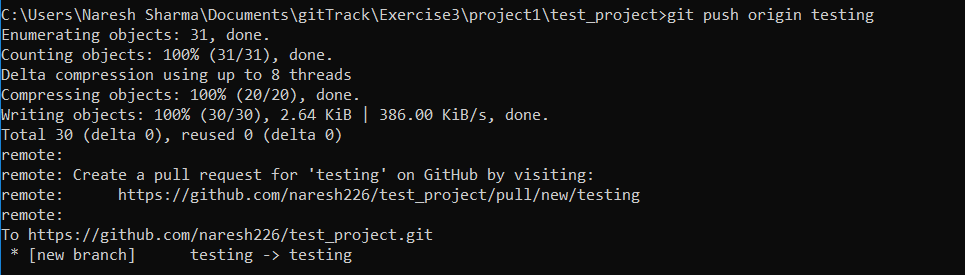




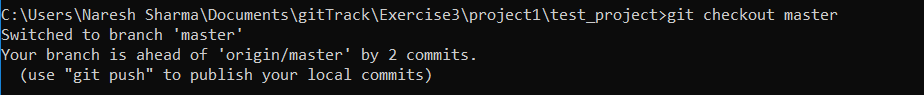




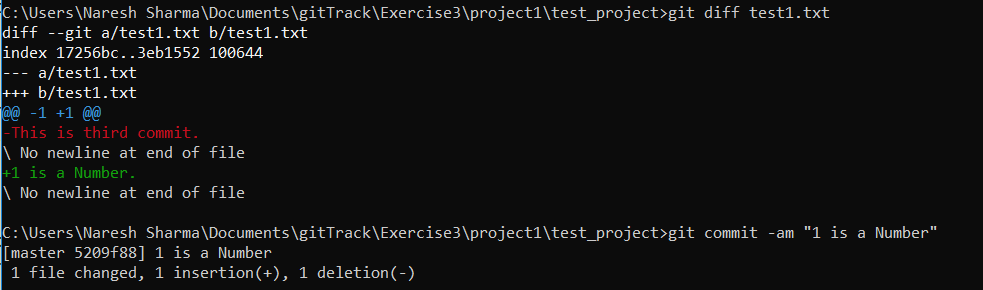
1. Push the branch to github.



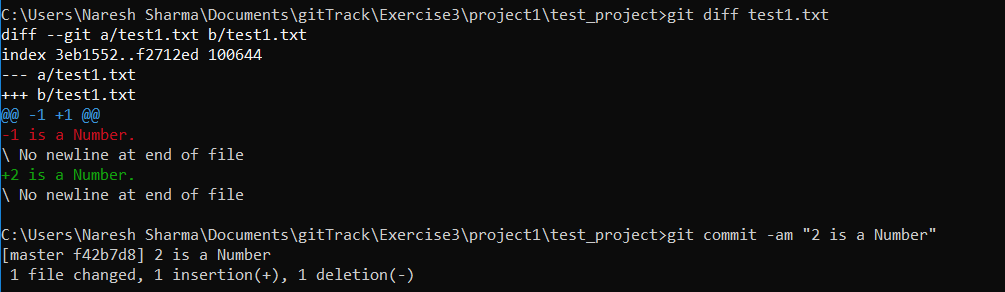
1. Checkout to master.

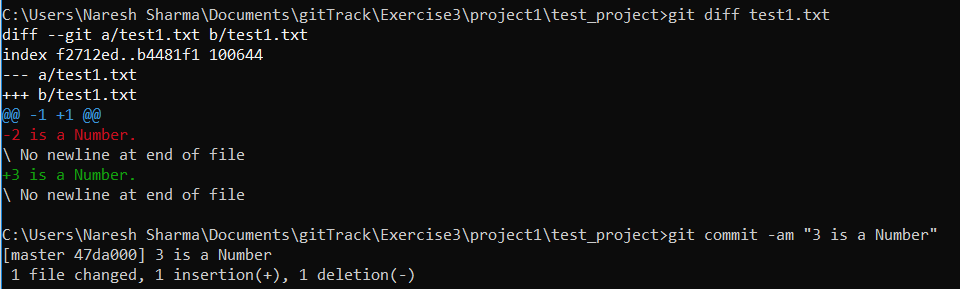


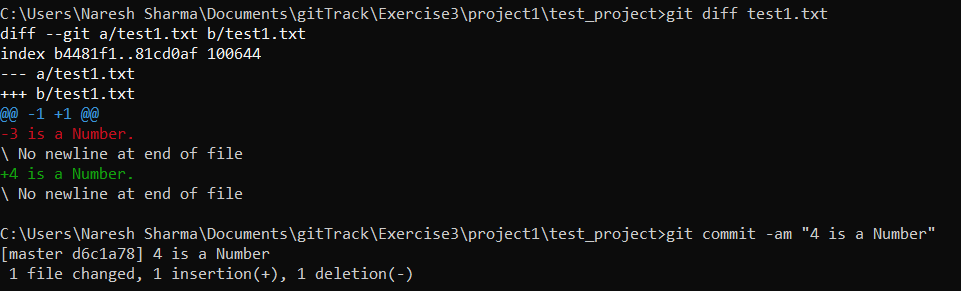
1. Change the text '1 is a number' and commit.

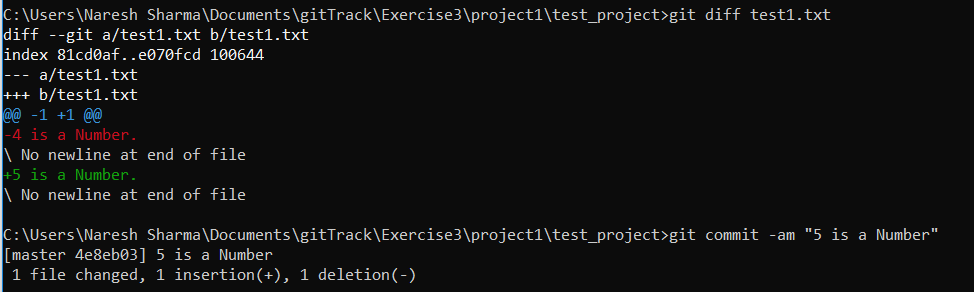


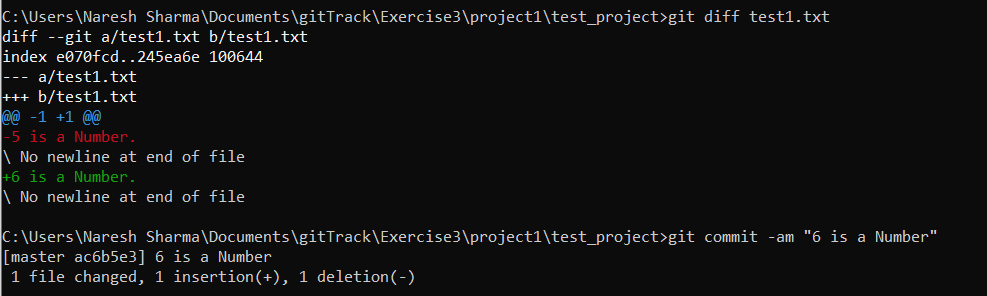
1. Repeat the 'step 22' seven times and each time update the number '1' to '2' to '3' etc.(Don't forget to commit the change everytime)

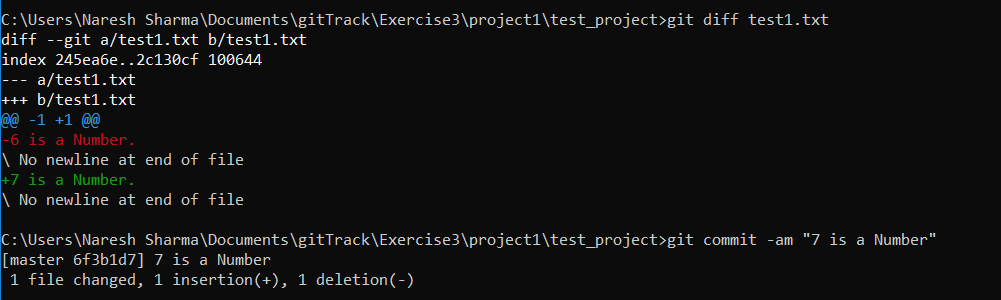


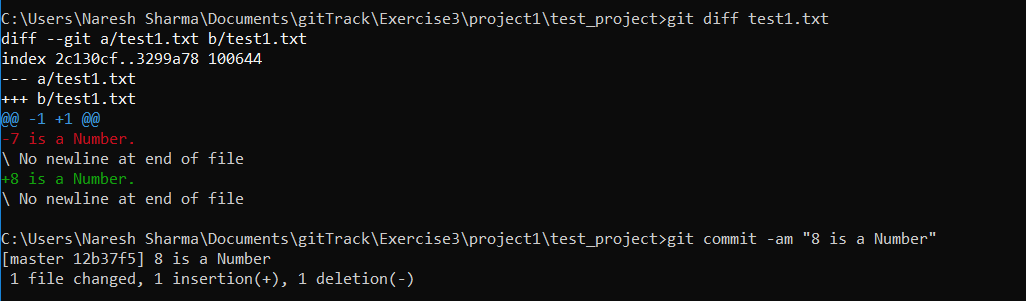




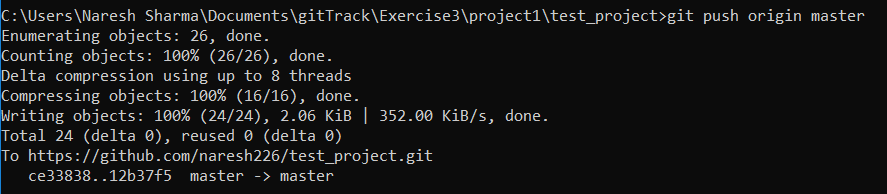




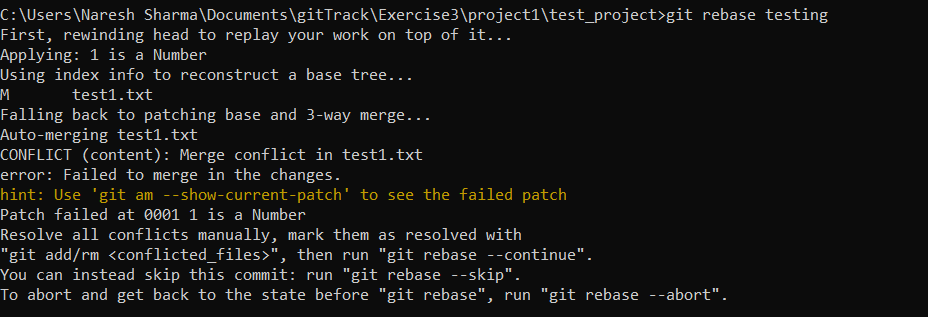


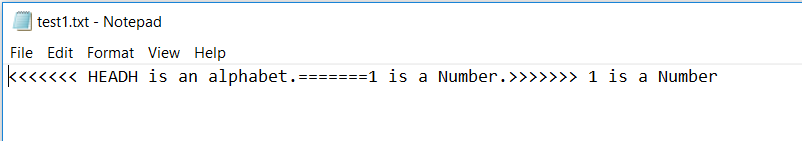


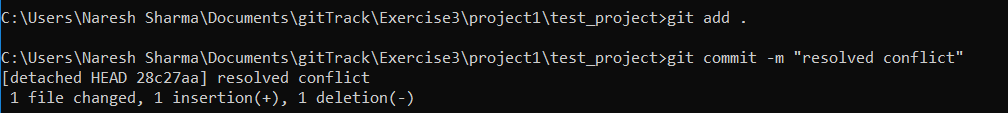
1. Push master branch to github.

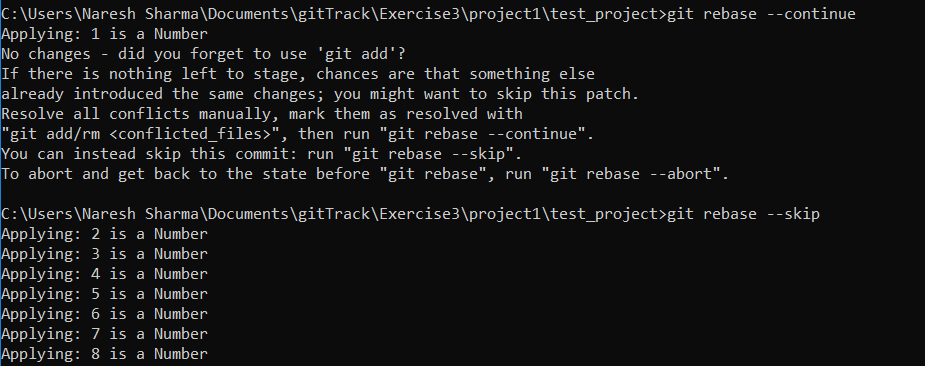


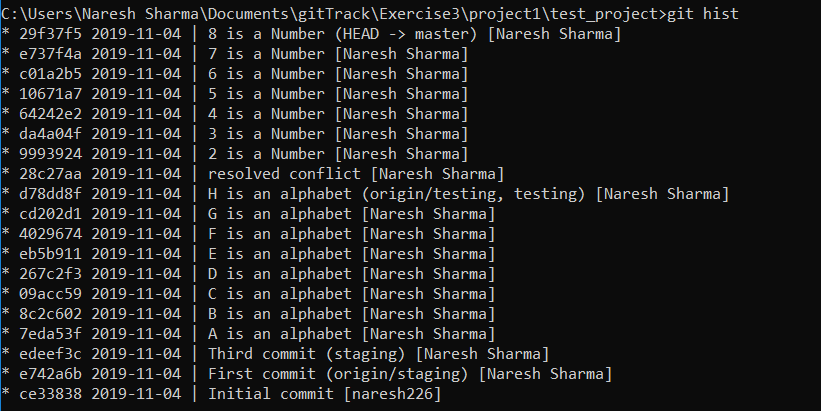
1. Now, rebase testing into master(and don't push the new master and testing branch).



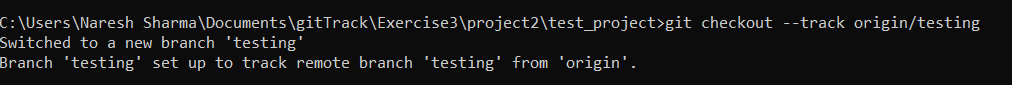




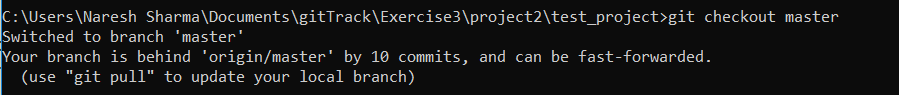




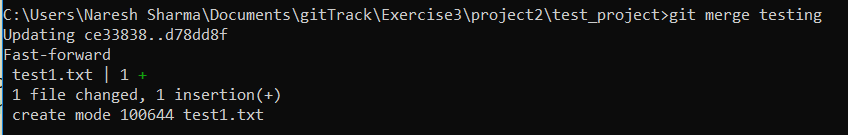
1. Move to project2.
2. Track testing branch.



1. Checkout to master.



1. Merge branch testing into master.



1. Share your views about the difference between master and testing.

Master branch is the default branch when we create a new repository and push changes to repository without specifying any branch whereas testing is new branch that is created from the master branch.

Changes in the master branch are combination of master and testing (when testing is merged with master) while the changes in the testing branch are specific to itself.

Testing branch is now tagged with a tracked branch which means that it will keep track of git push and pull from remote branch i.e it automatically push or pull changes from remote branch to which it is connected.

