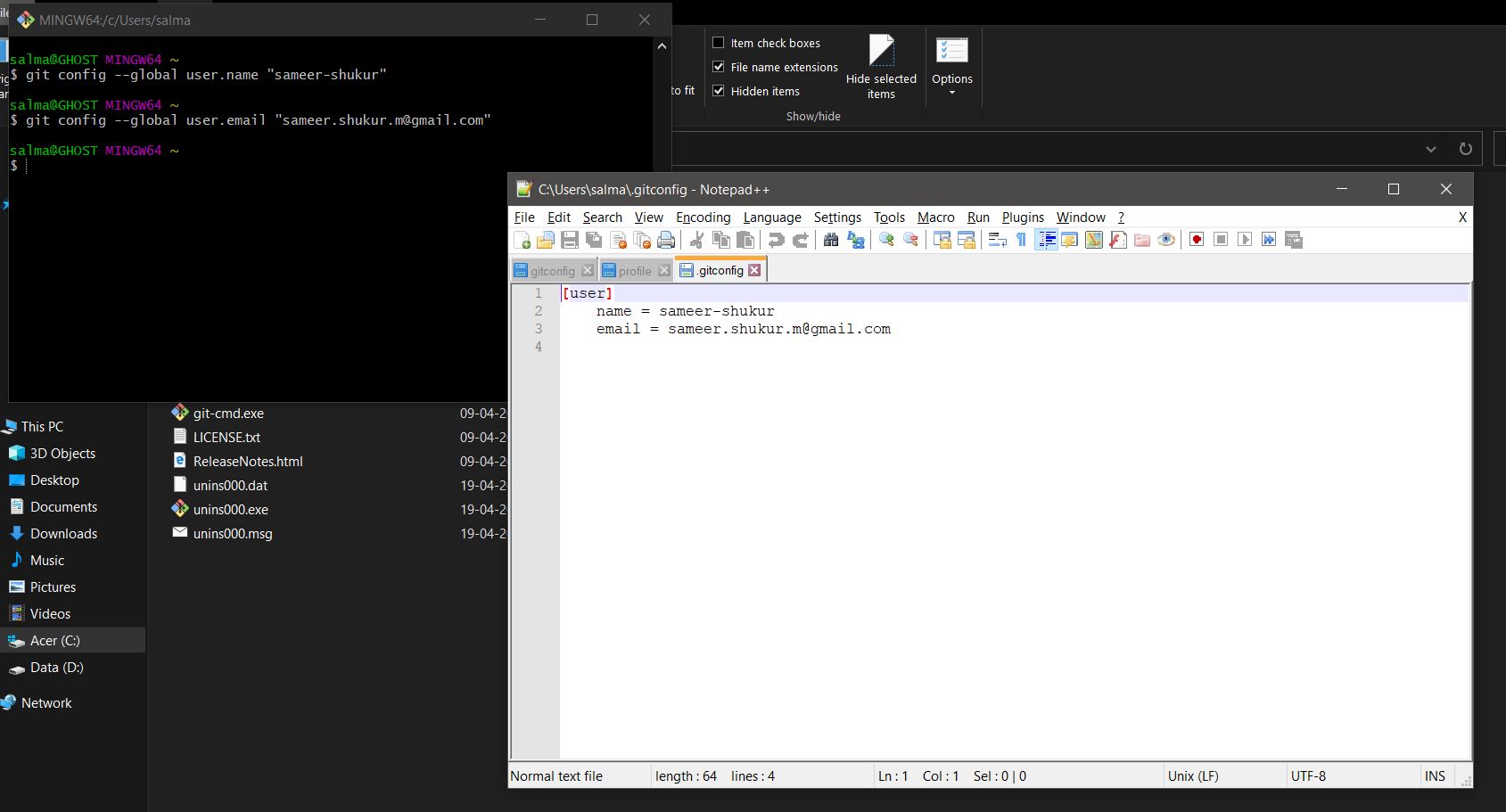
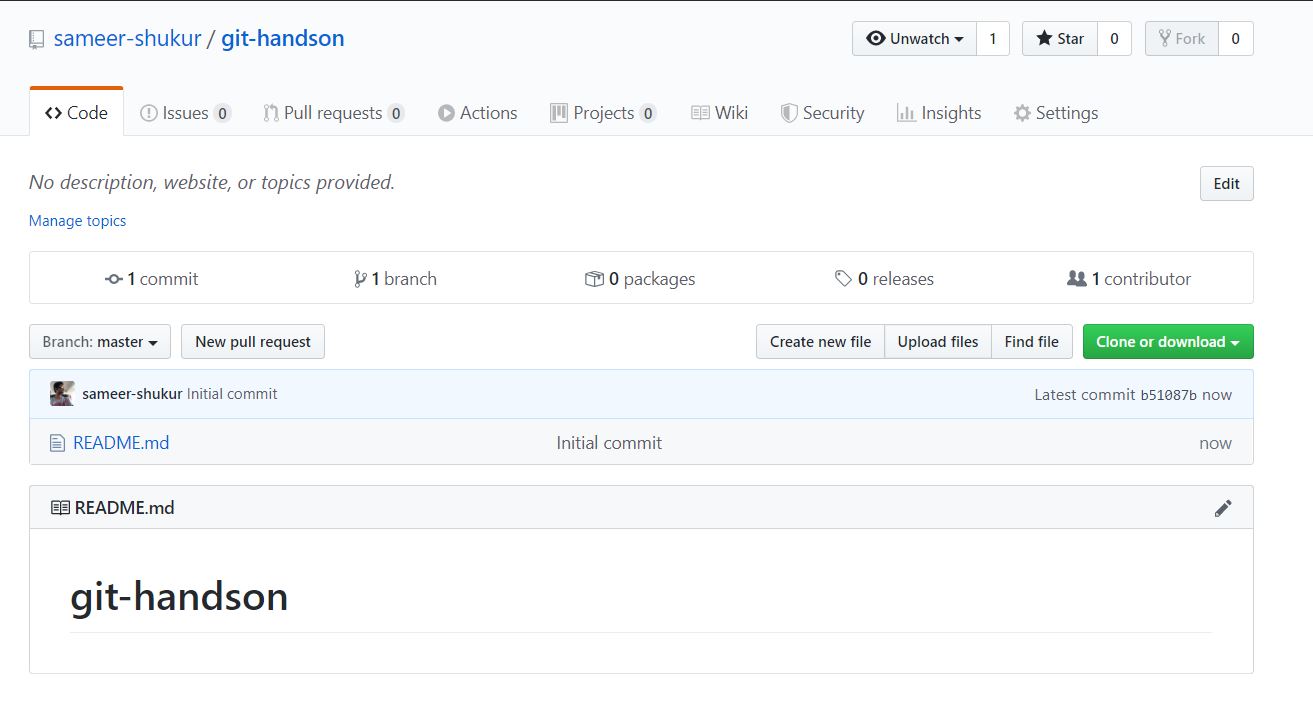
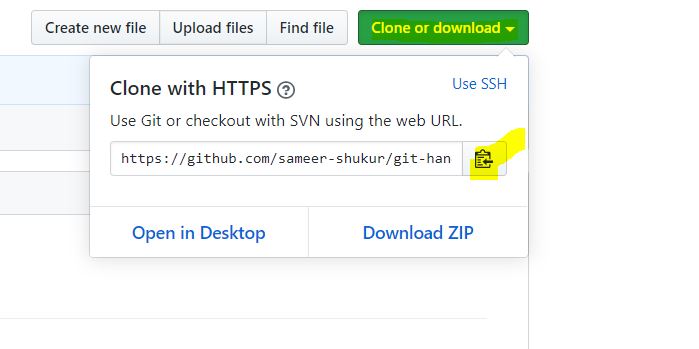
1. Set the global configuration file with your user name, email and editor as Notepad++. List all the properties which you just set.



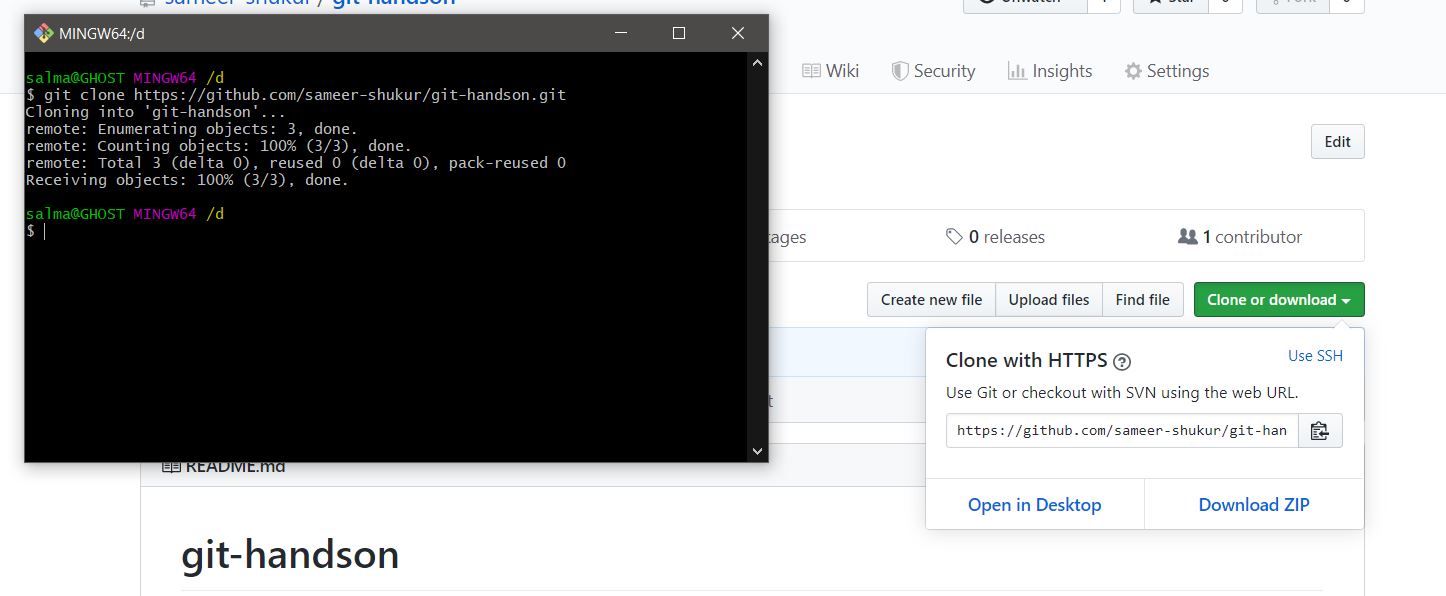
1. Make a fresh Git project



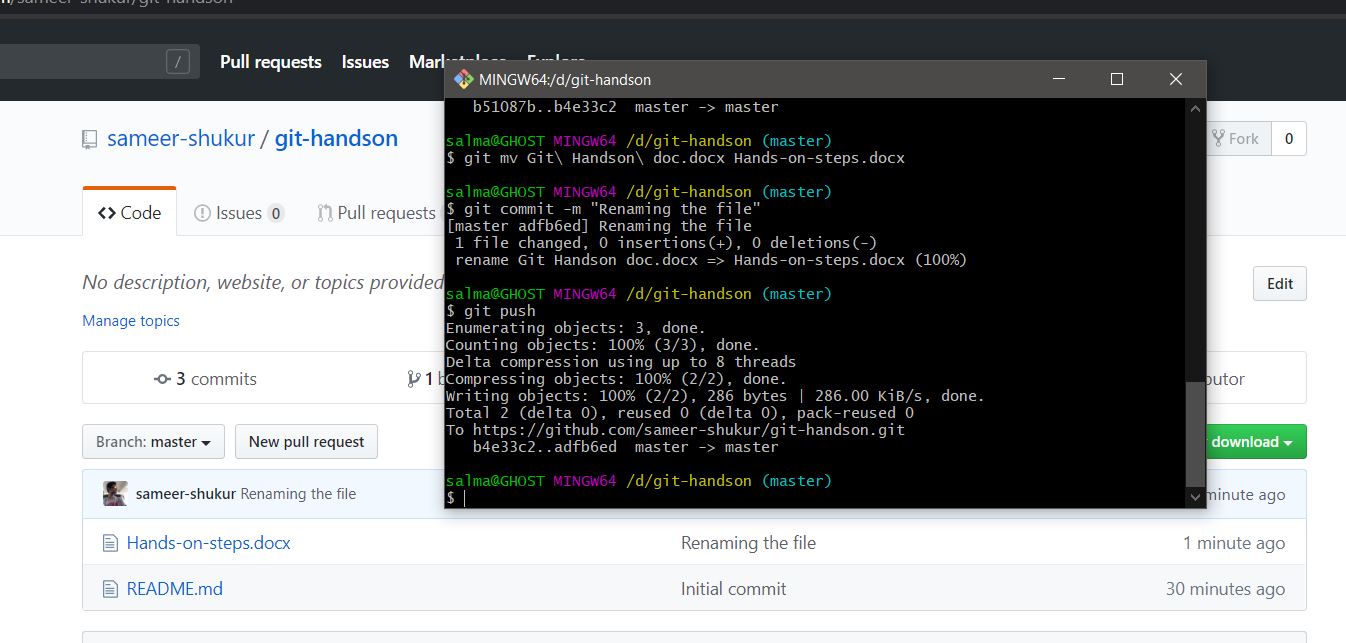
1. Using existing Github account and cloning the above created projected to local repository.



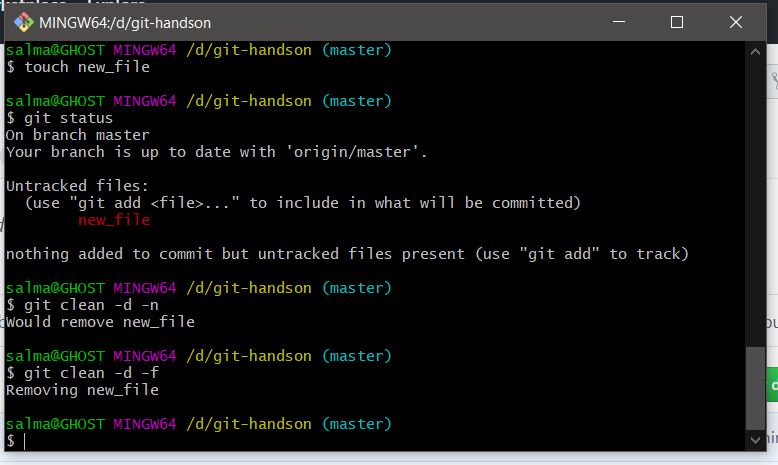
1. Push the project to remote repo



1. Use the different ways of renaming and moving files



1. You just created a new file, but then you decided that the file is to be removed. How do you delete this untracked file.

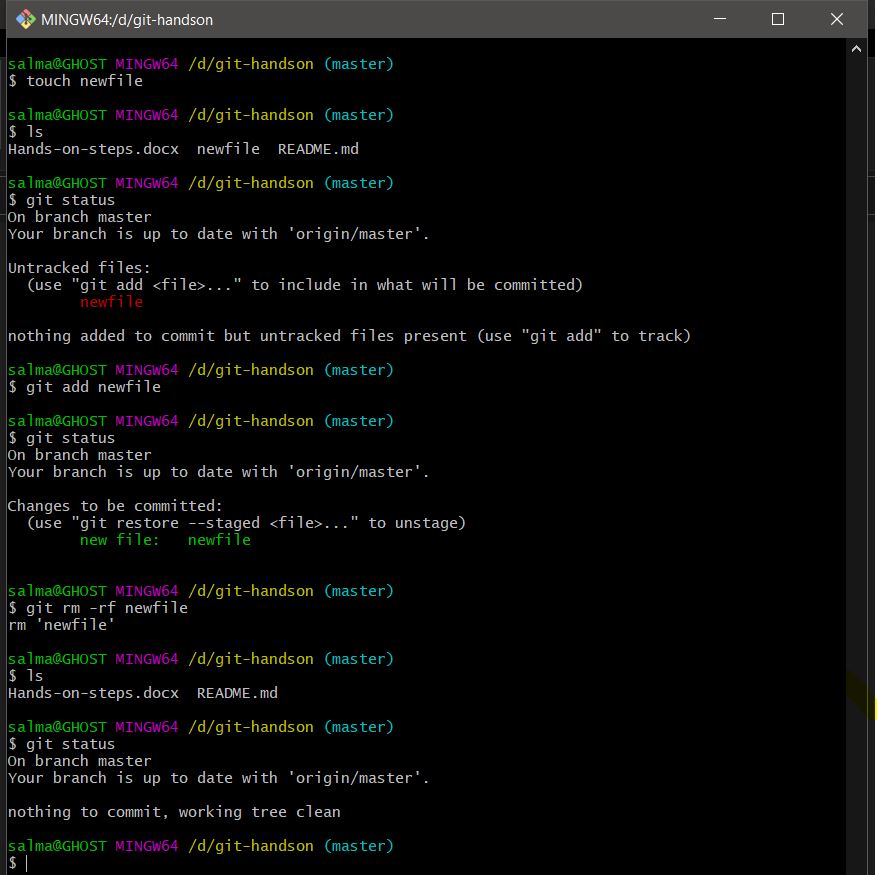
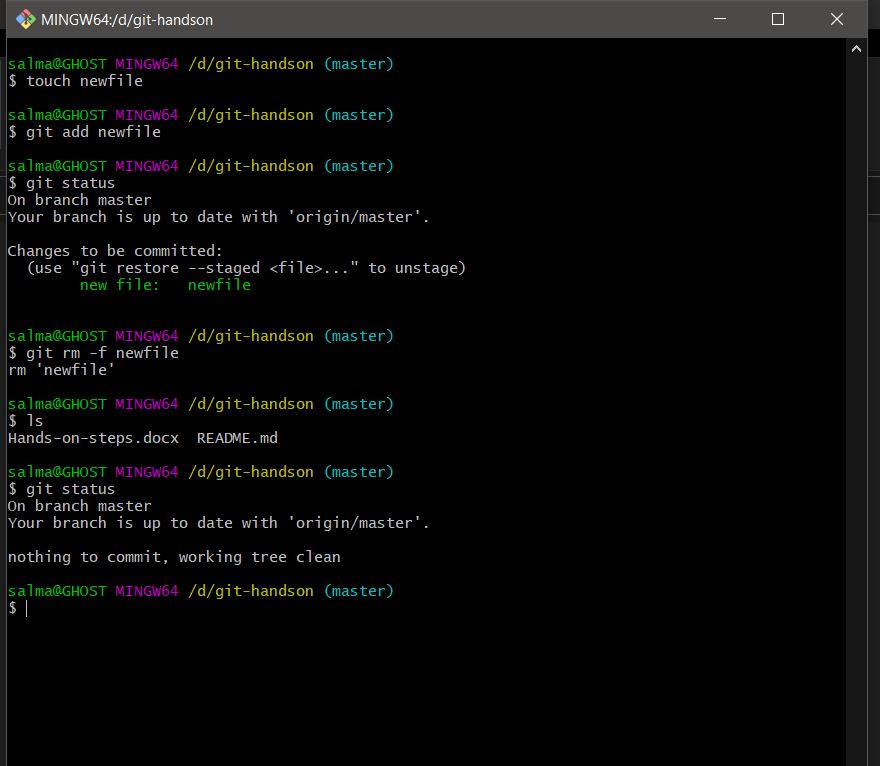


7. Demonstrate the following:

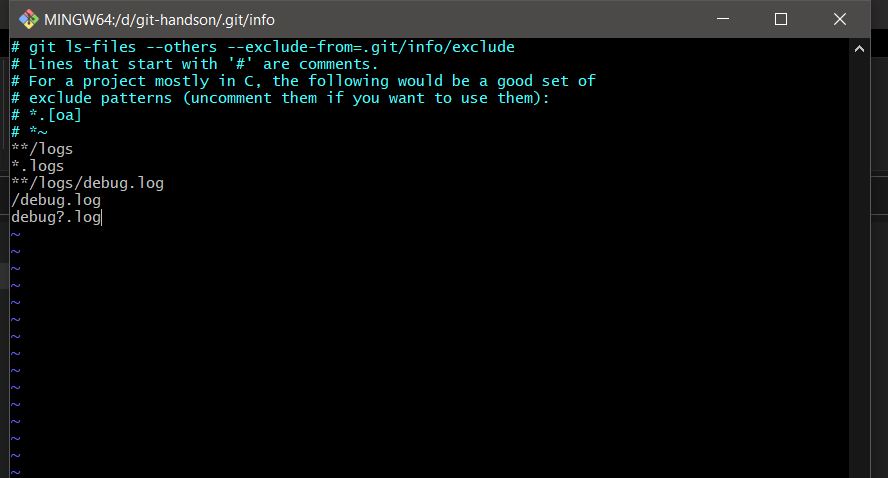
a. delete of a tracked file

b. backing out staged deletion

c. recursive deletion



8. You do not want to push certain folders/files of your project; how do you manage this?



9. Create a branch called “test”. Make some changes in the master branch. Let there be some changes in the working directory and some in the staging area. Make some changes in the test branch as well. Issue the command to show the differences for

a. Working directory vs Staging area

b. Working directory vs Local Repository

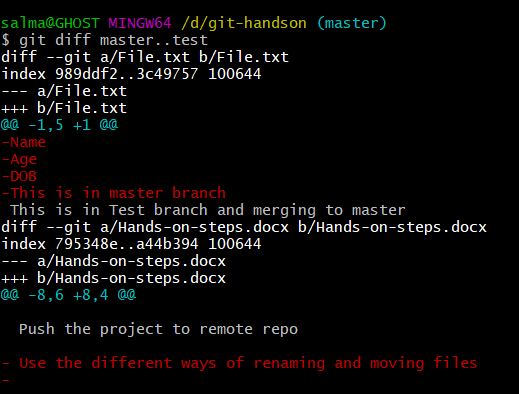
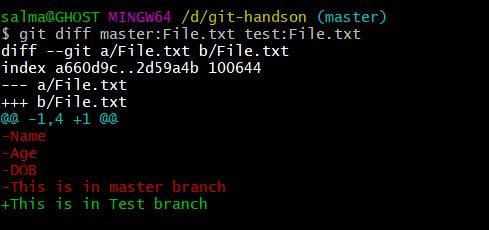
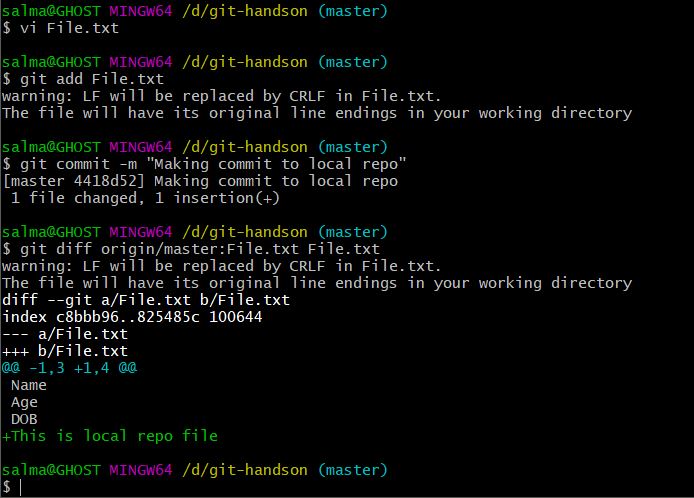
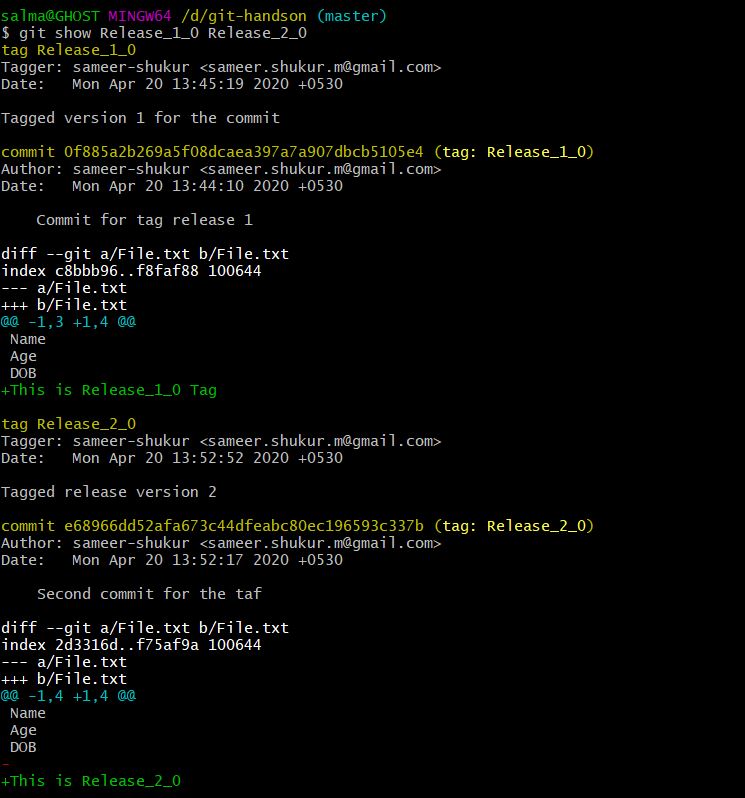
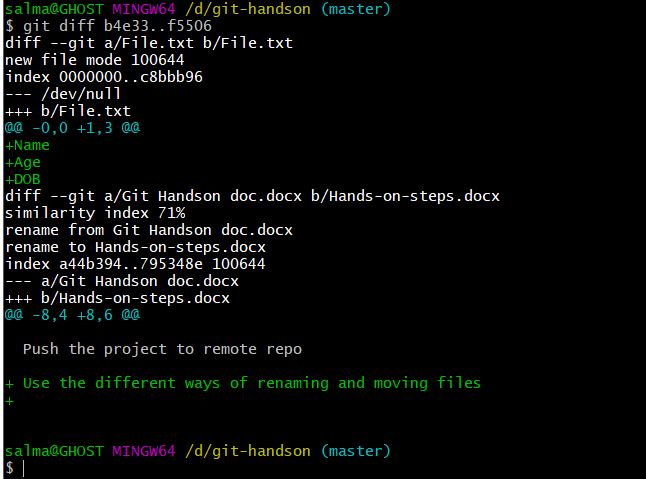
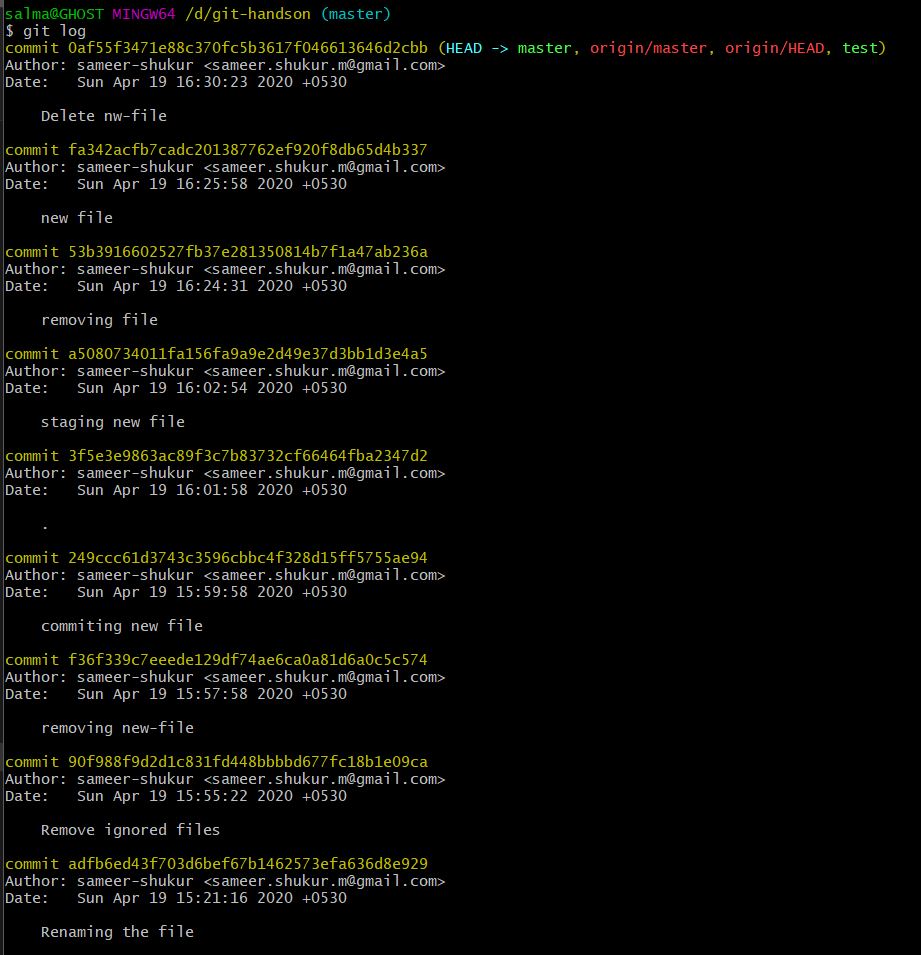
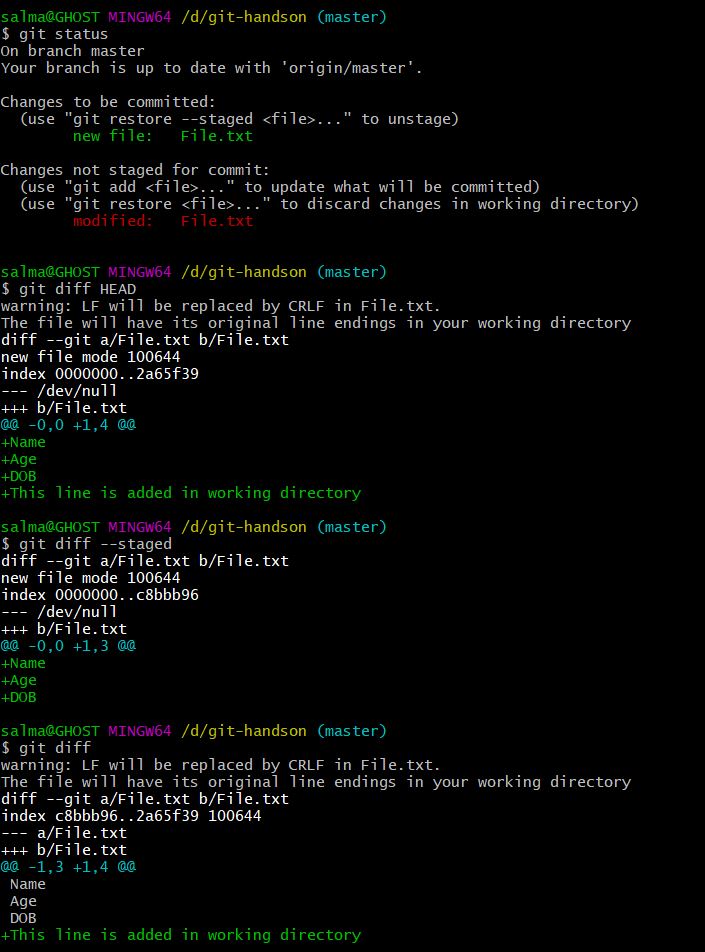
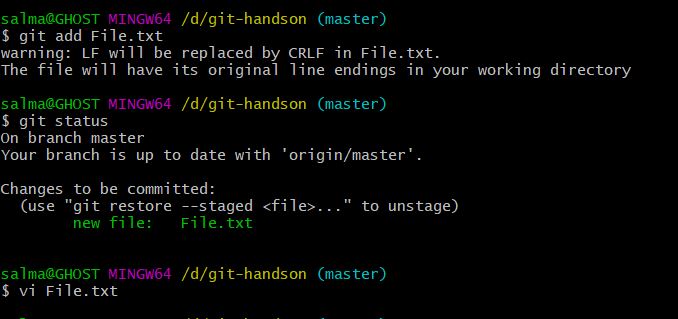
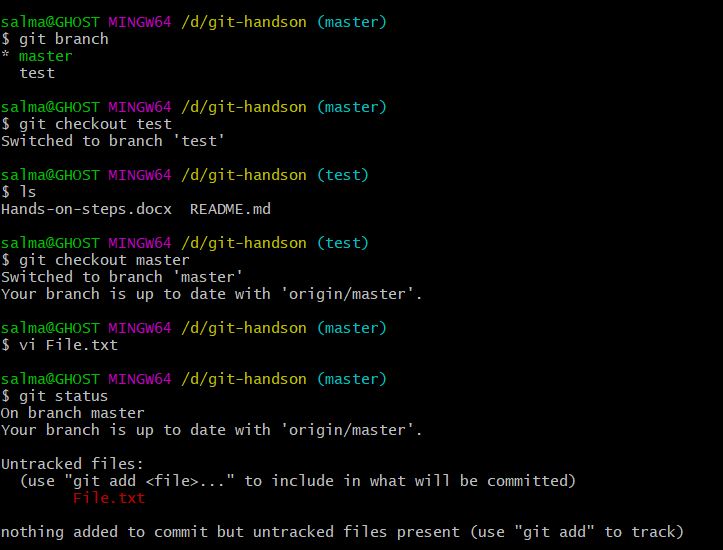
c. Staging area vs Local Repository

d. Between two commits

e. Between two tags

f. Local vs Remote Repository

g. Master branch vs test branch



10. Merge the changes from test branch to master branch.

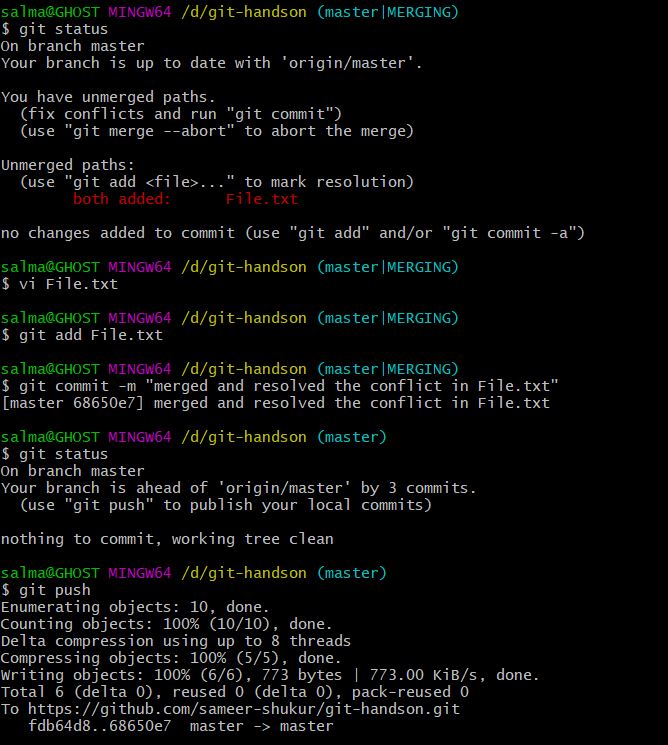
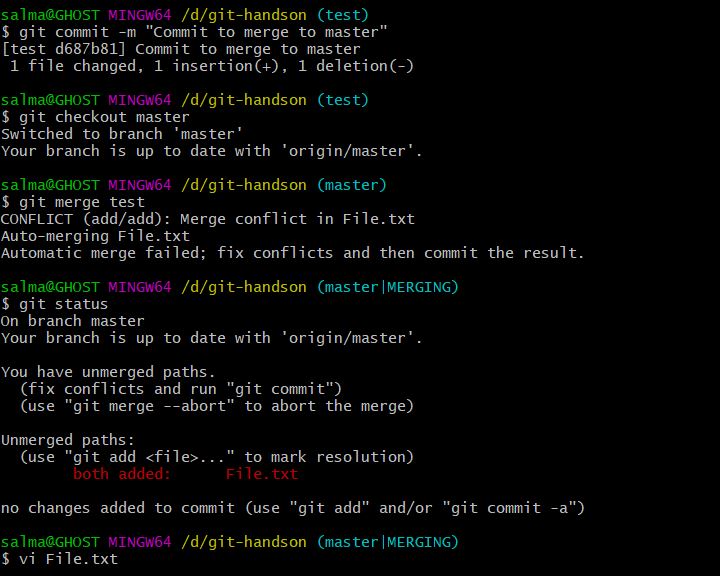
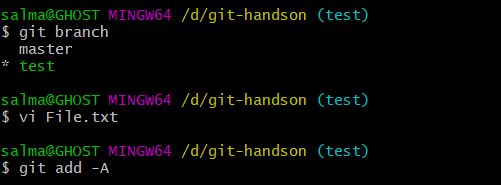
a. FastForward merge

b. Disabling FastForward merge

c. What is the difference between option a and option b

* I was not able to perform this task, as I was facing merge conflict issue, tried multiple times.

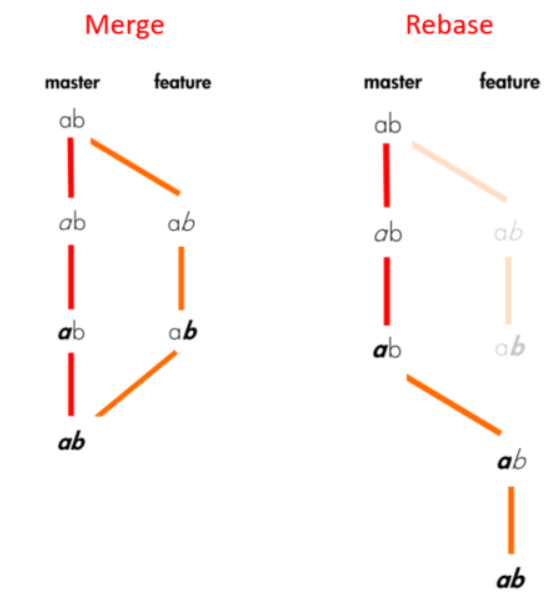
11. Create a merge conflict situation. Resolve the conflict and merge the changes between the branches.



12. What is the difference between merge and rebase, demonstrate with the eg.

**git-merge** - Join two or more development histories together.

**git-rebase** – Sequentially regenerate a series of commits so they can be applied directly to the head node.



13. With an example, demonstrate fetch, clone and pull. What is the usecase for these operations. Are they same, different? Explain

**Git Pull:** Git pull will pull down from a remote whatever you ask for and instantly merge it into the branch you’re in when you make the request. Pull is a high-level request that runs ‘fetch’ then a ‘merge’ by default.

%> git checkout localBranch

%> git pull origin master

%> git branch

master

\* localBranch

The above will merge the remote “master” branch into the local “localBranch”.

**Git Fetch:** Fetch is similar to pull, except it won’t do any merging.

%> git checkout localBranch

 %> git fetch origin remoteBranch

%> git branch

master

\* localBranch

remoteBranch

So, the fetch will have pulled down the remoteBranch. Creates a local copy of a remote branch which you shouldn’t manipulate directly; instead create a proper local branch and work on that. ‘git checkout’ has a confusing feature though. If you ‘checkout’ a local copy of a remote branch, it creates a local copy and sets up a merge to it by default.

**Git Clone:** Git clone will clone a repo into a newly created directory.

%> cd git-handson

%> git clone https://github.com/sameer-shukur/git-handson.git

%> git branch

 \* master

test

14. Create a new repository in Github, with a README file. While pushing to the remote repository, if the remote branch is ahead of the local repository (new file is added in remote repository, which is not there in local repository) and pull is failing, how do you solve this problem?

- Rebasing the code with the one in the master and then update our changes on top of it will resolve the issue.