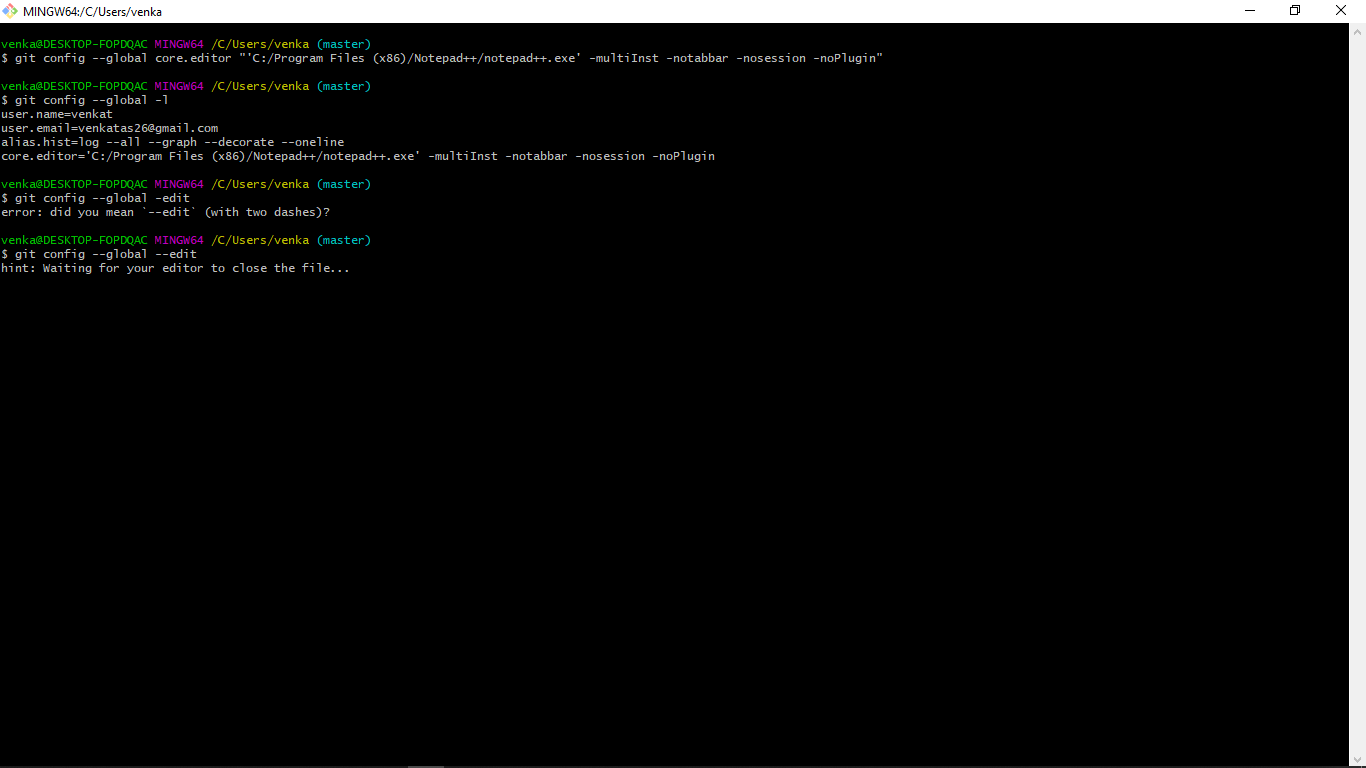
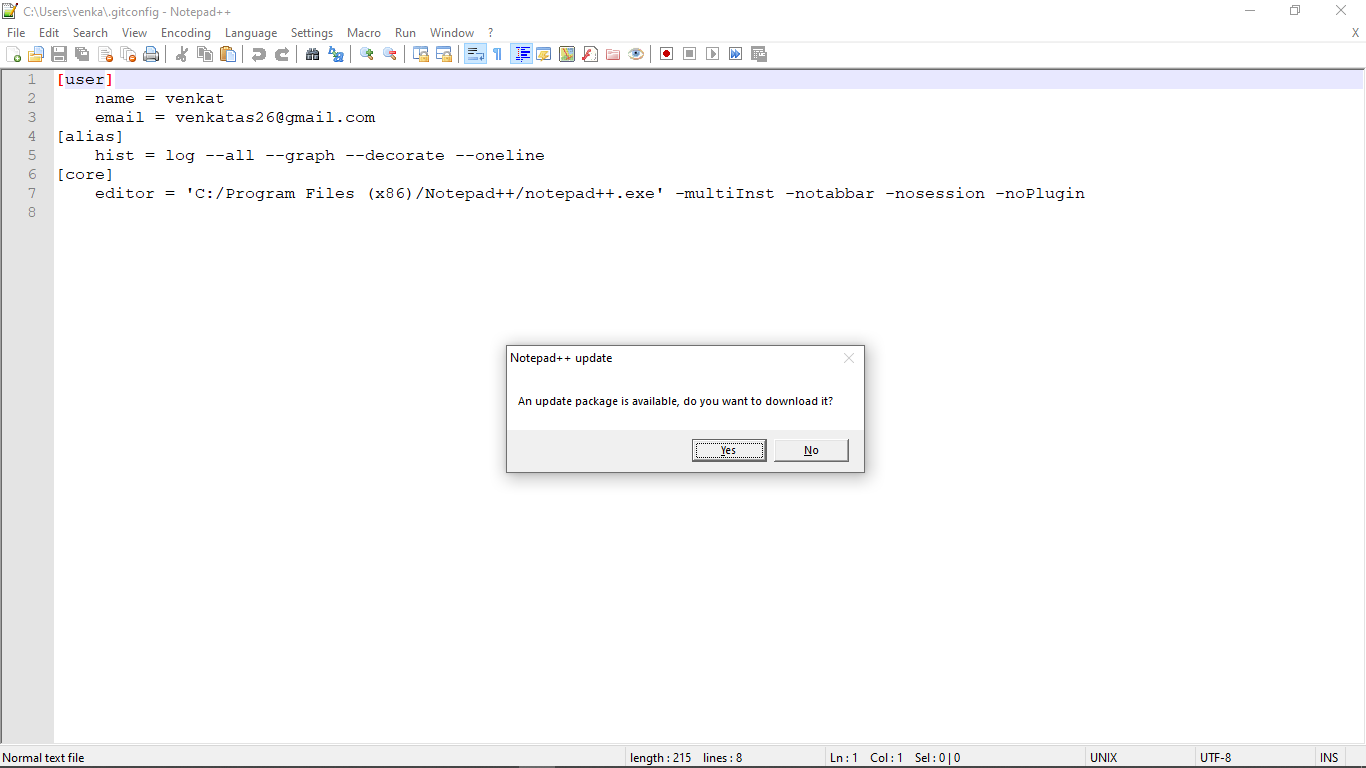
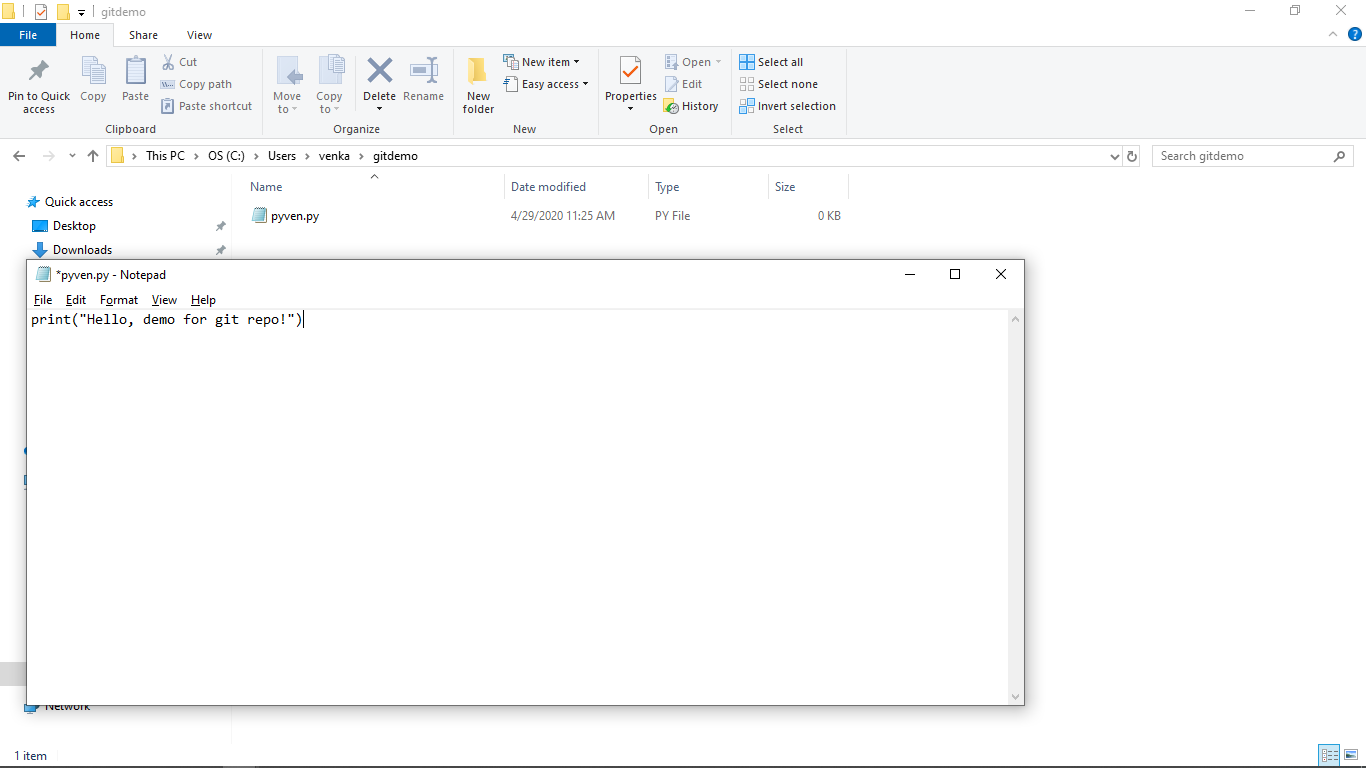
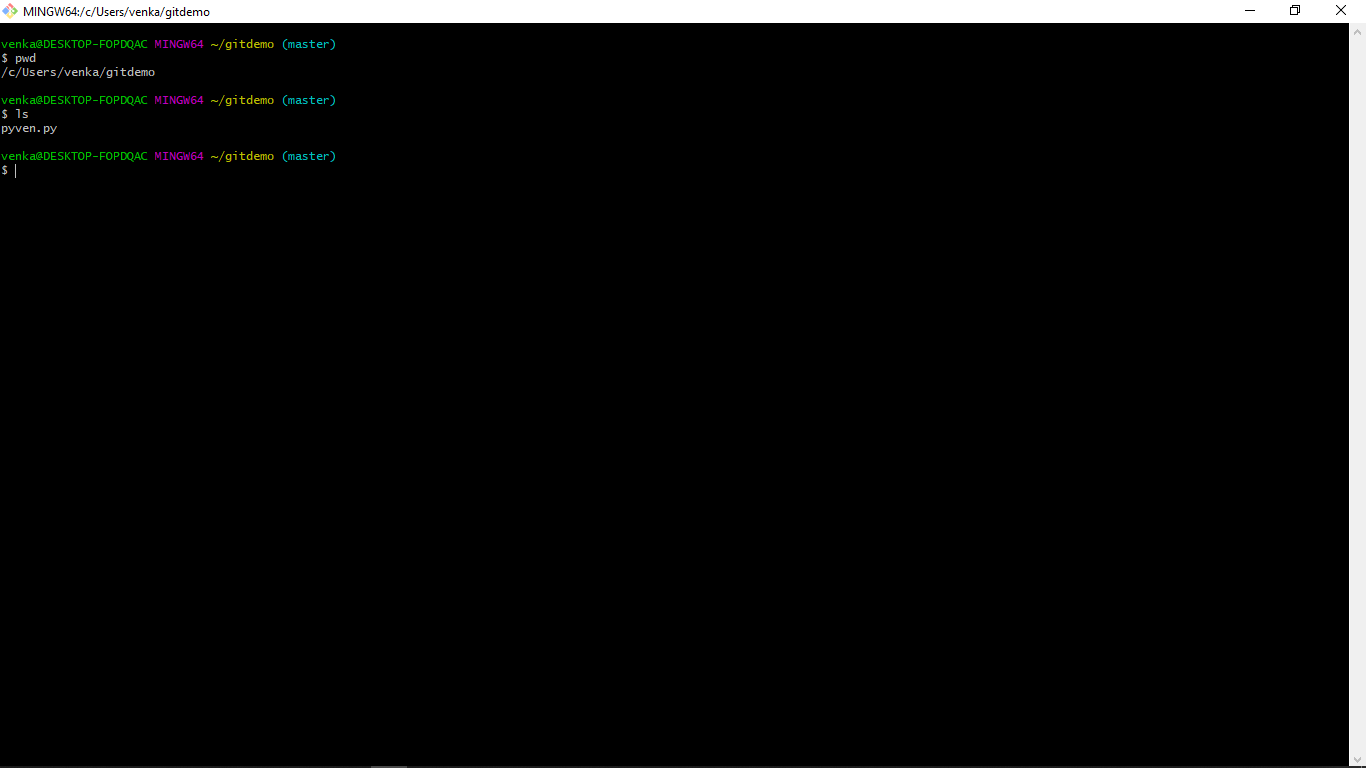
1.

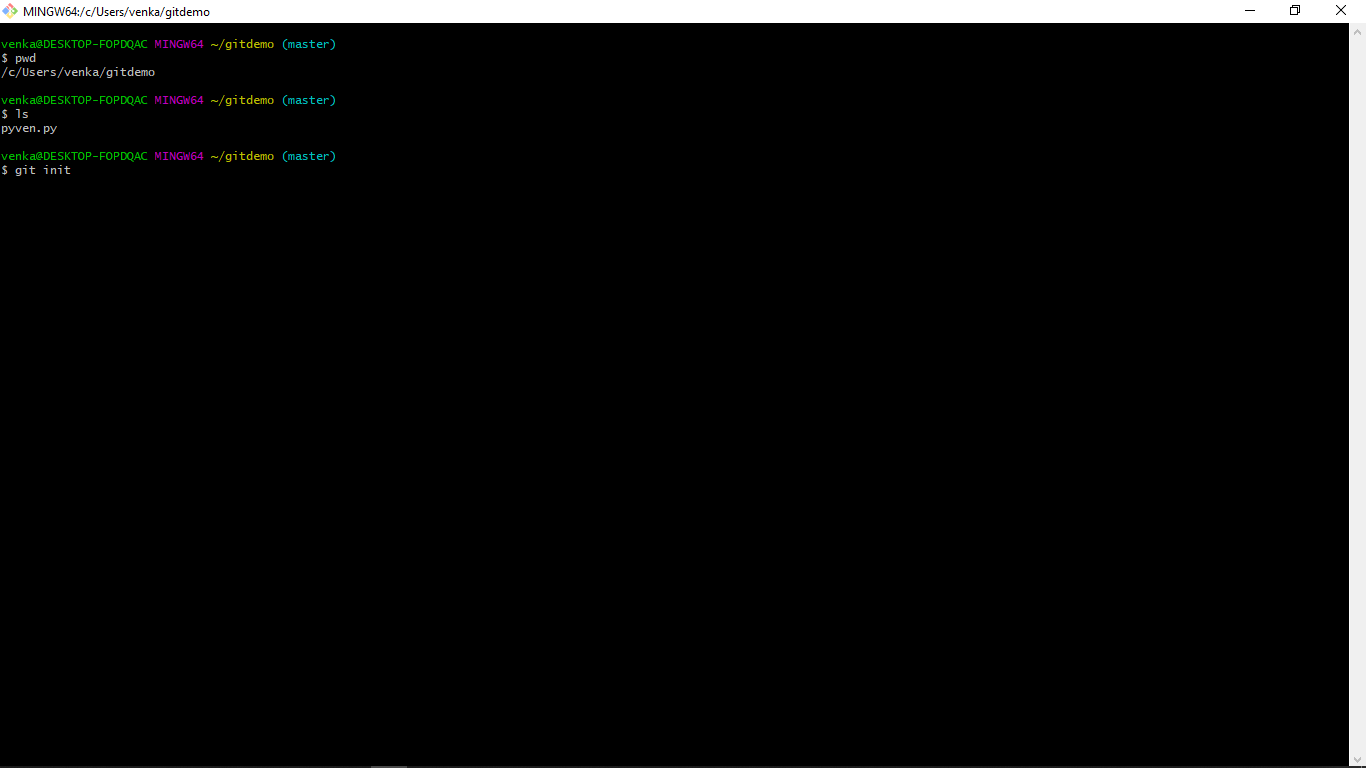


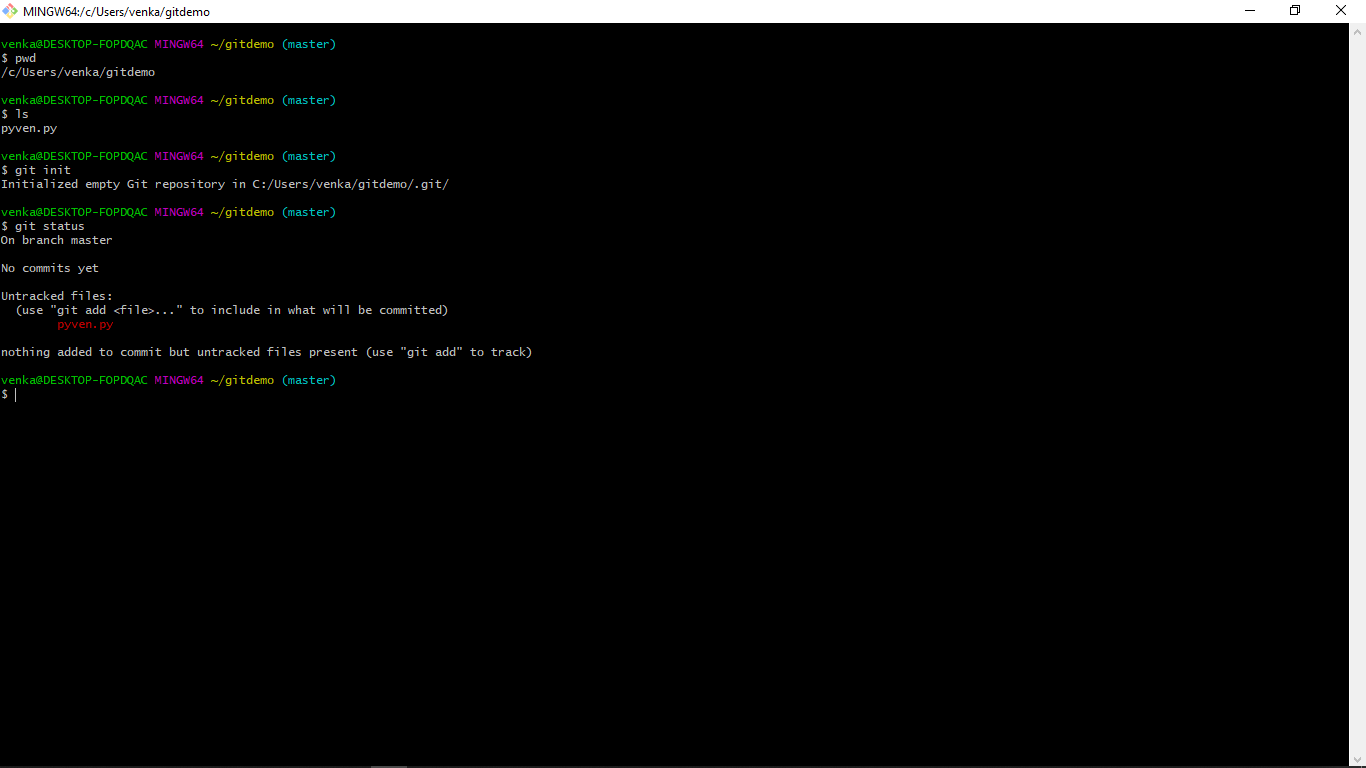


2. Created a directory for a small python project as shown below and attaching it to local git as shown below:

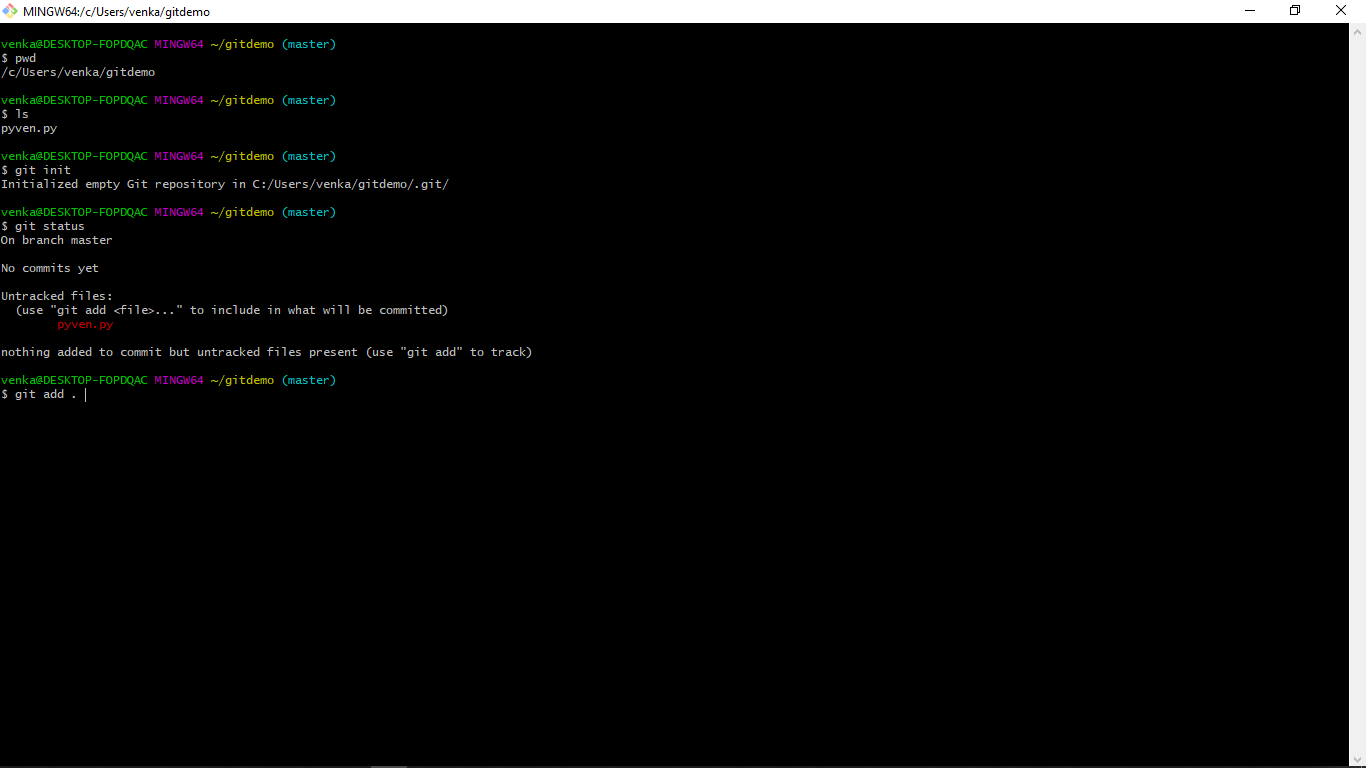


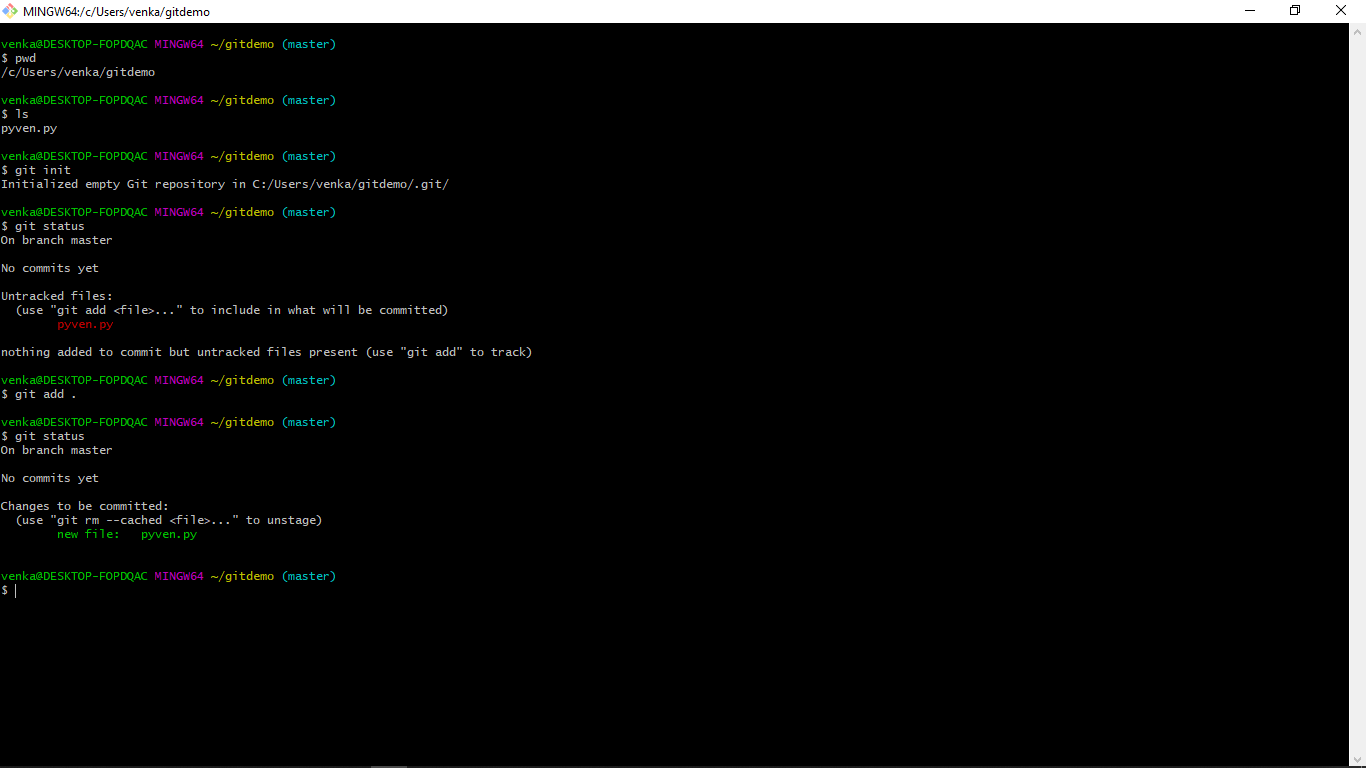




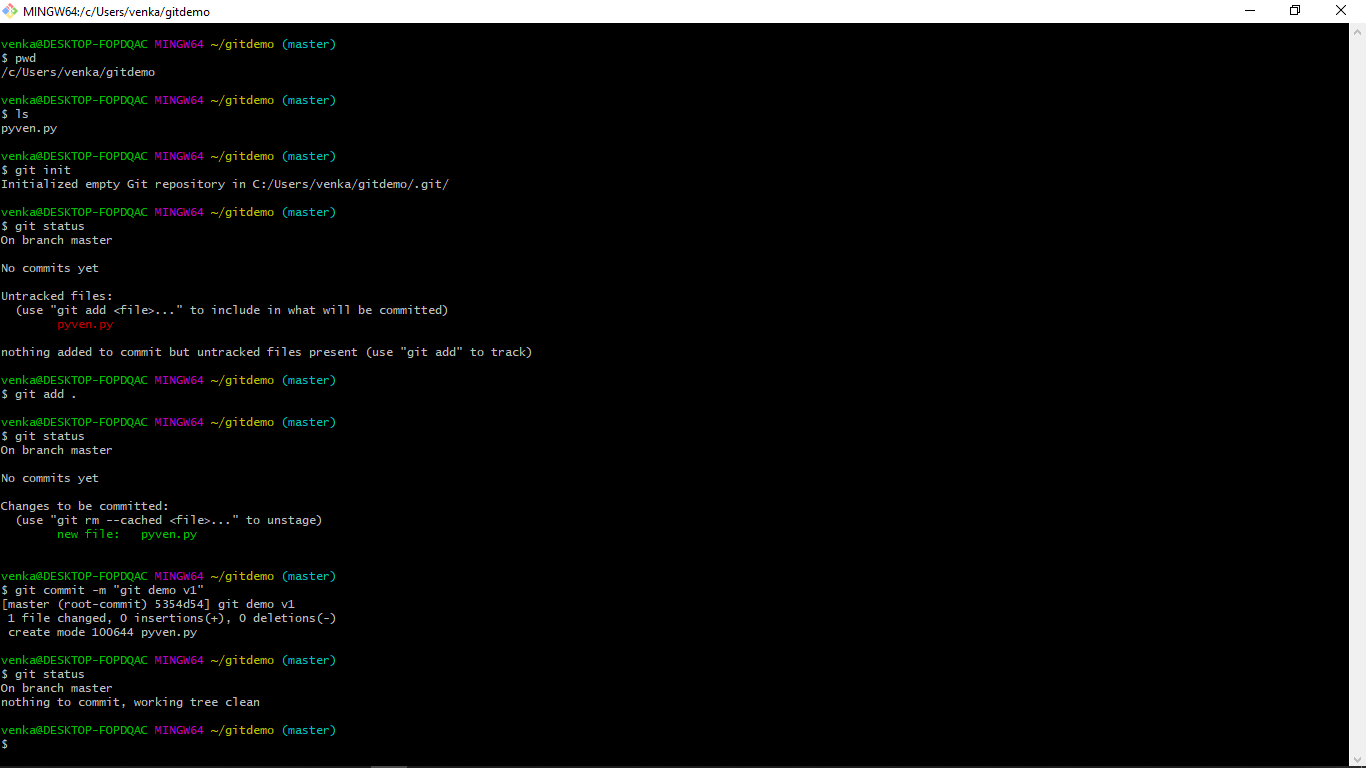


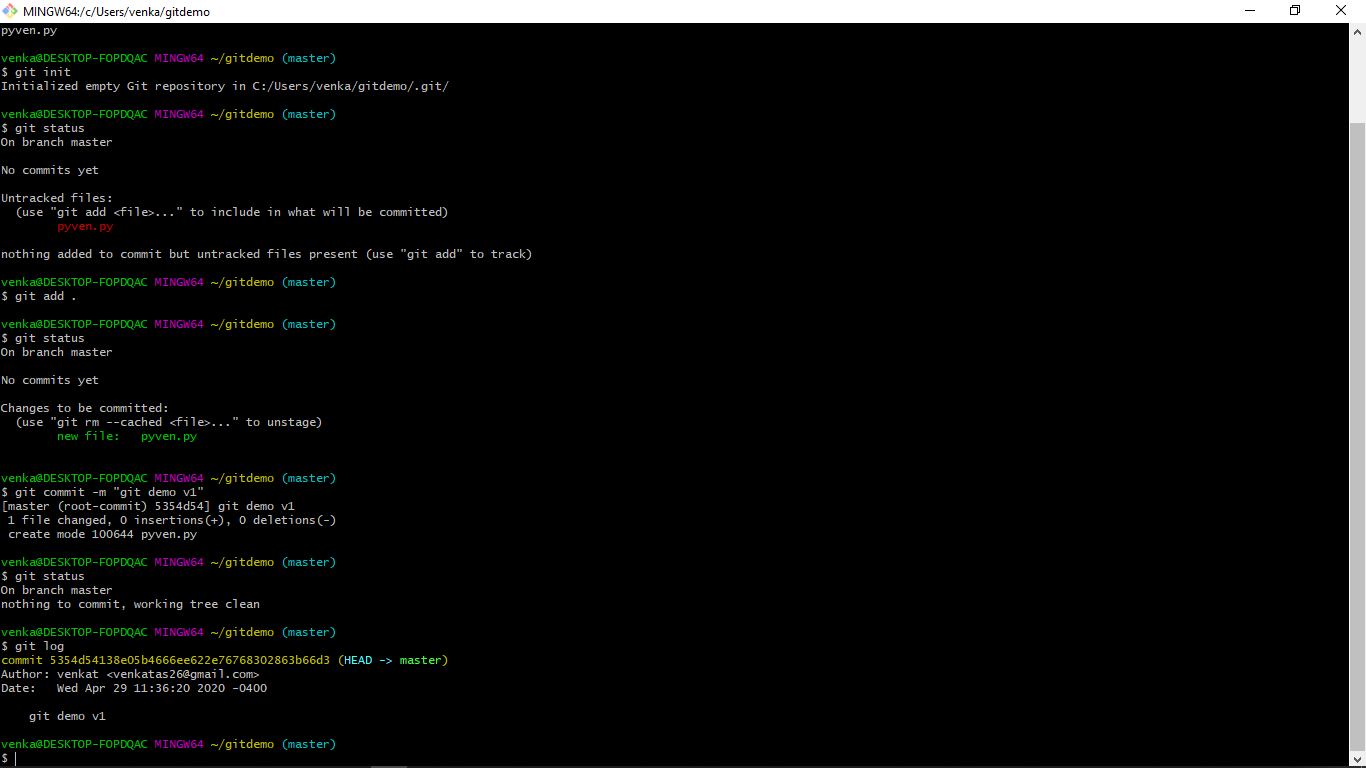
Staged the changes as shown below using git add.



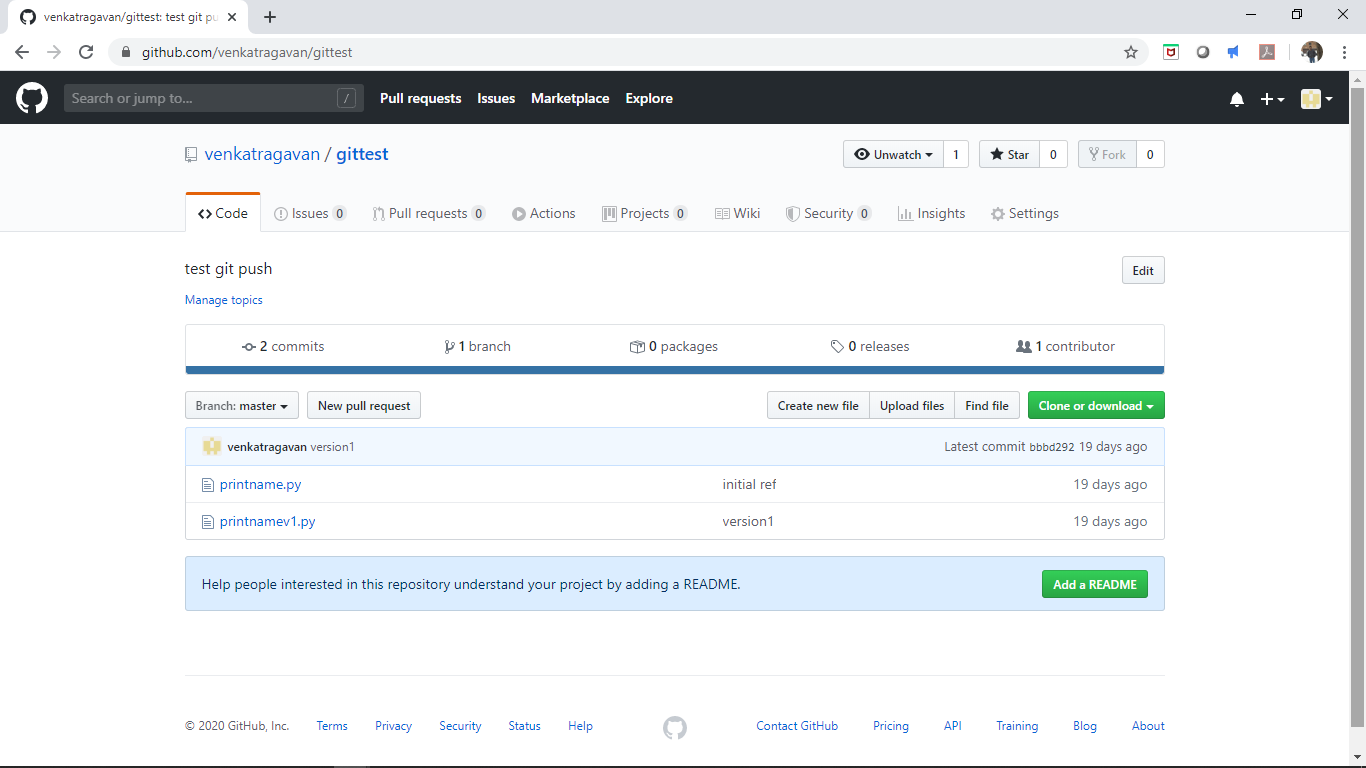


Will commit the changes to the git repository using git commit

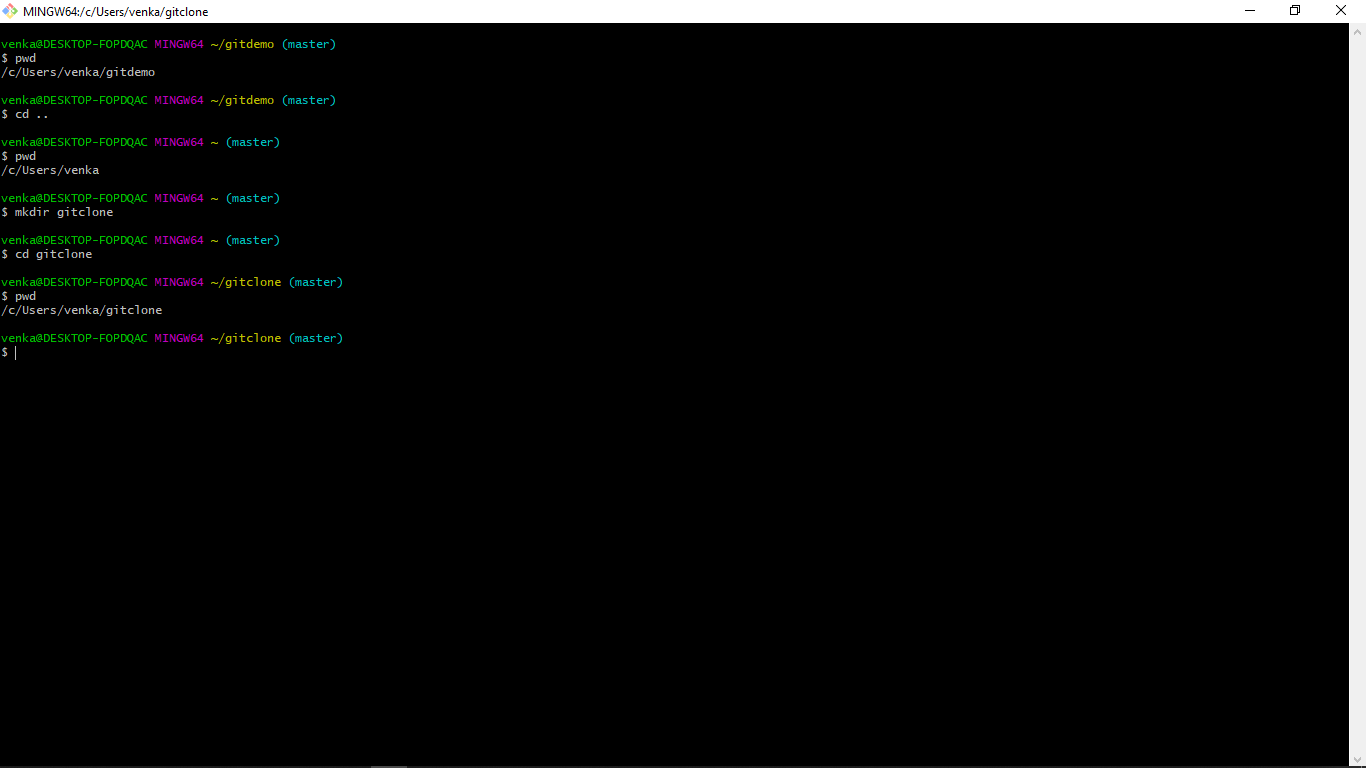




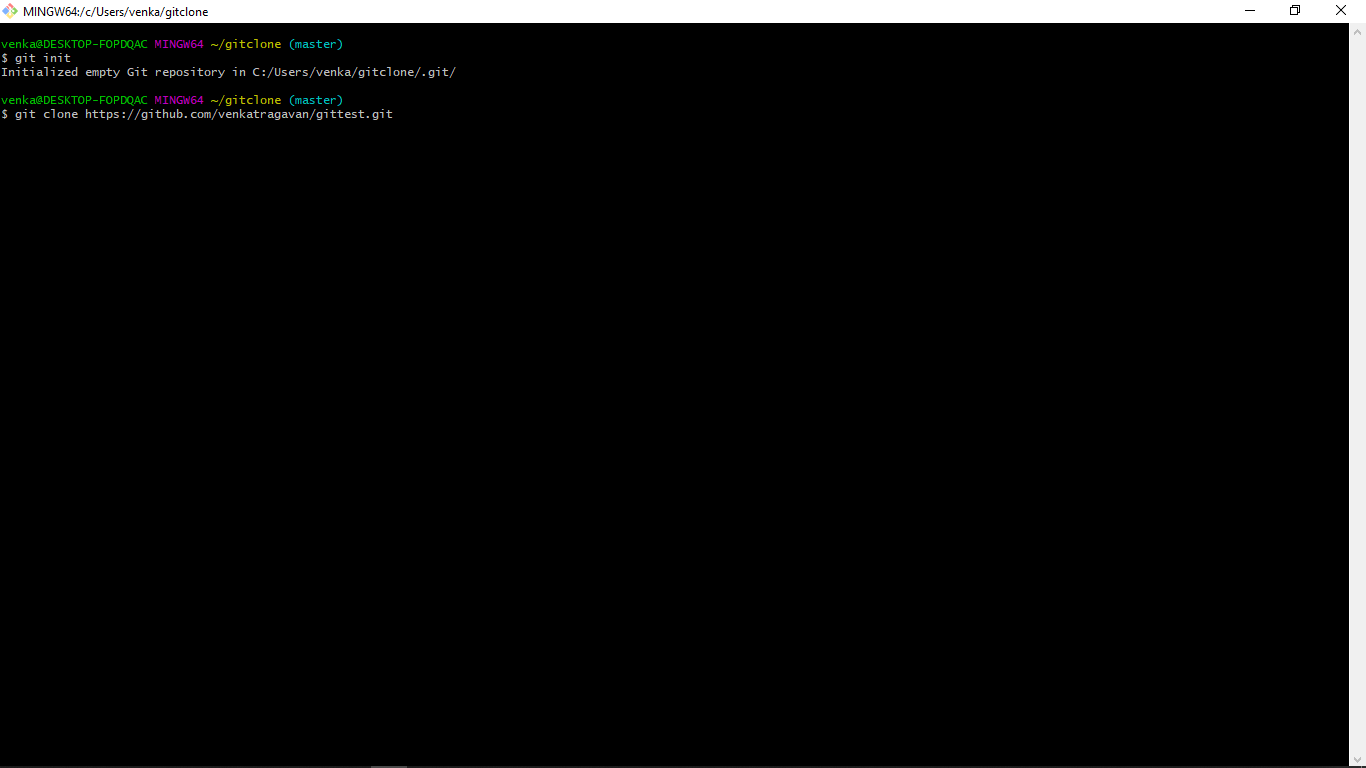
3. My Github account has a small python project as shown belwo:

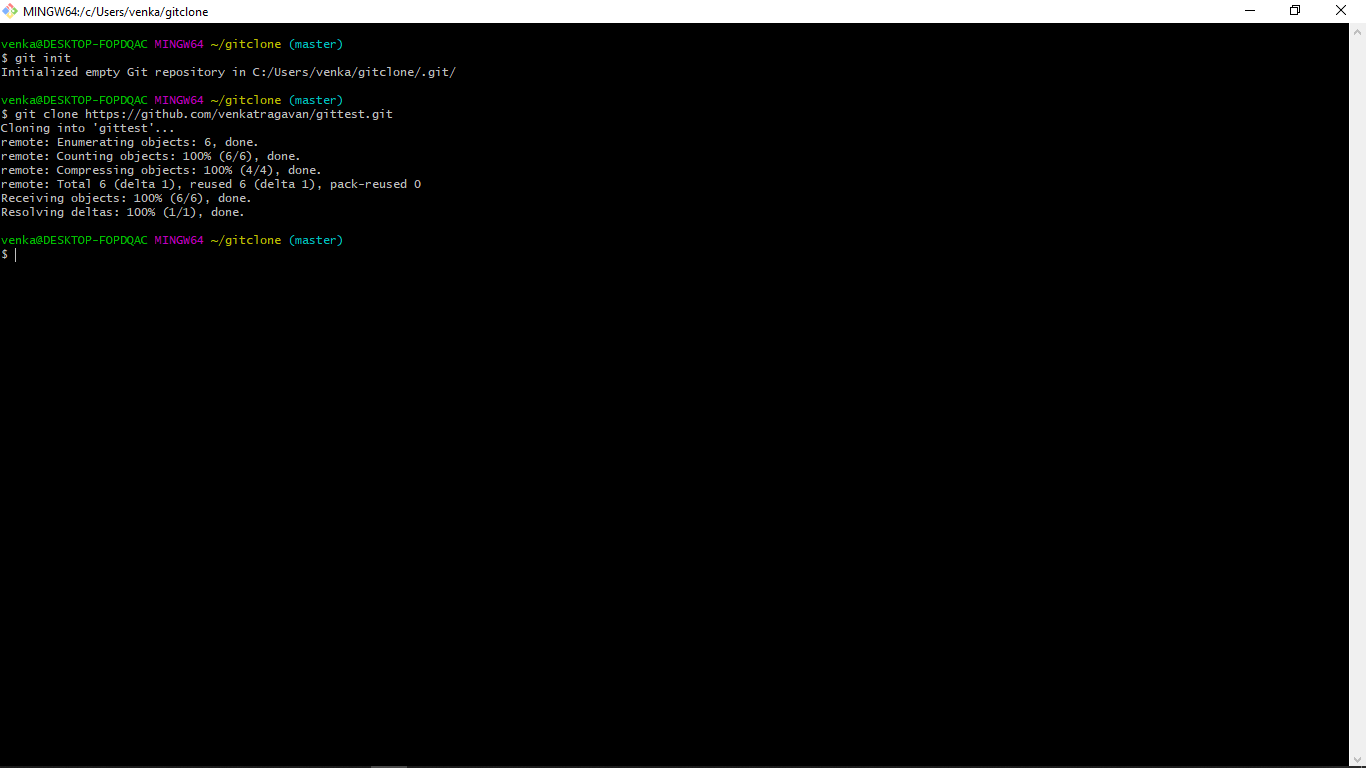


Created a new directory in local git repo as shown below:

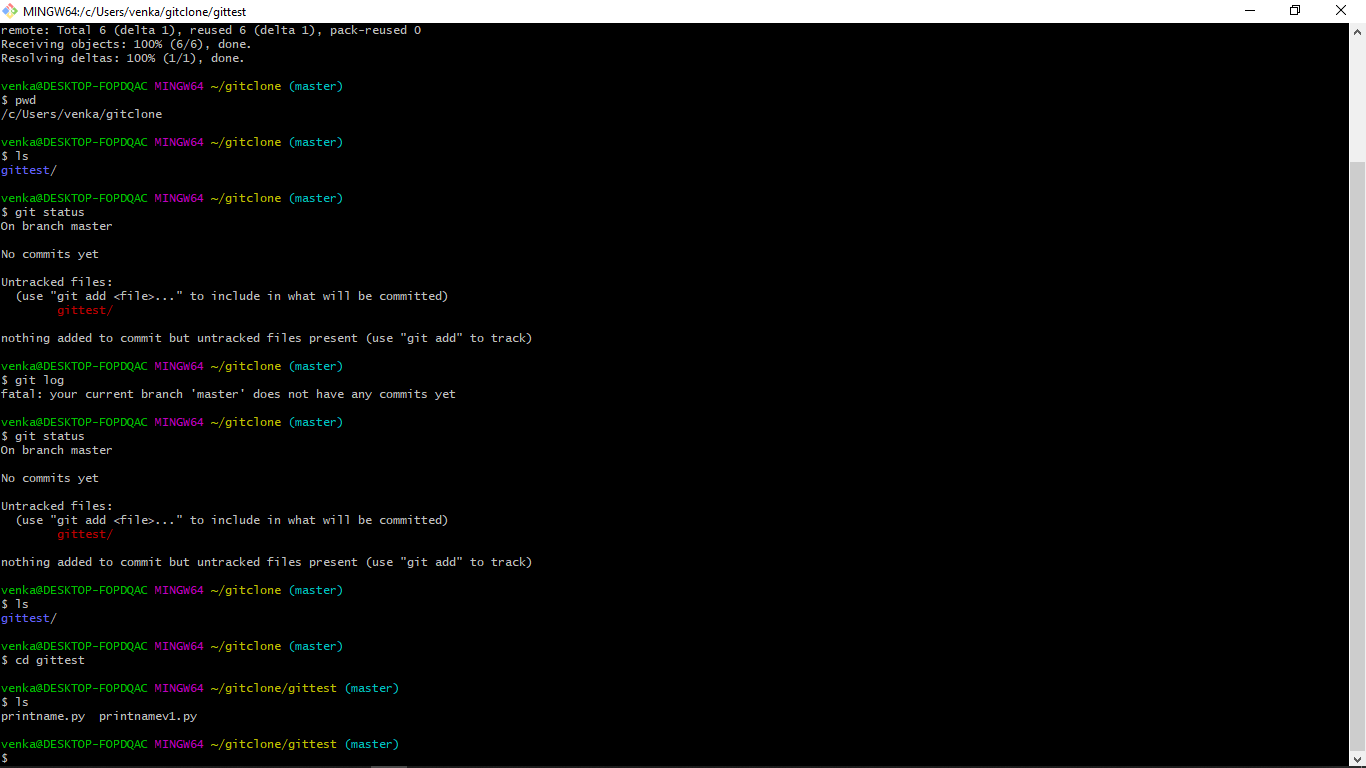


Cloned the remote repo URL to the local git as shown below:

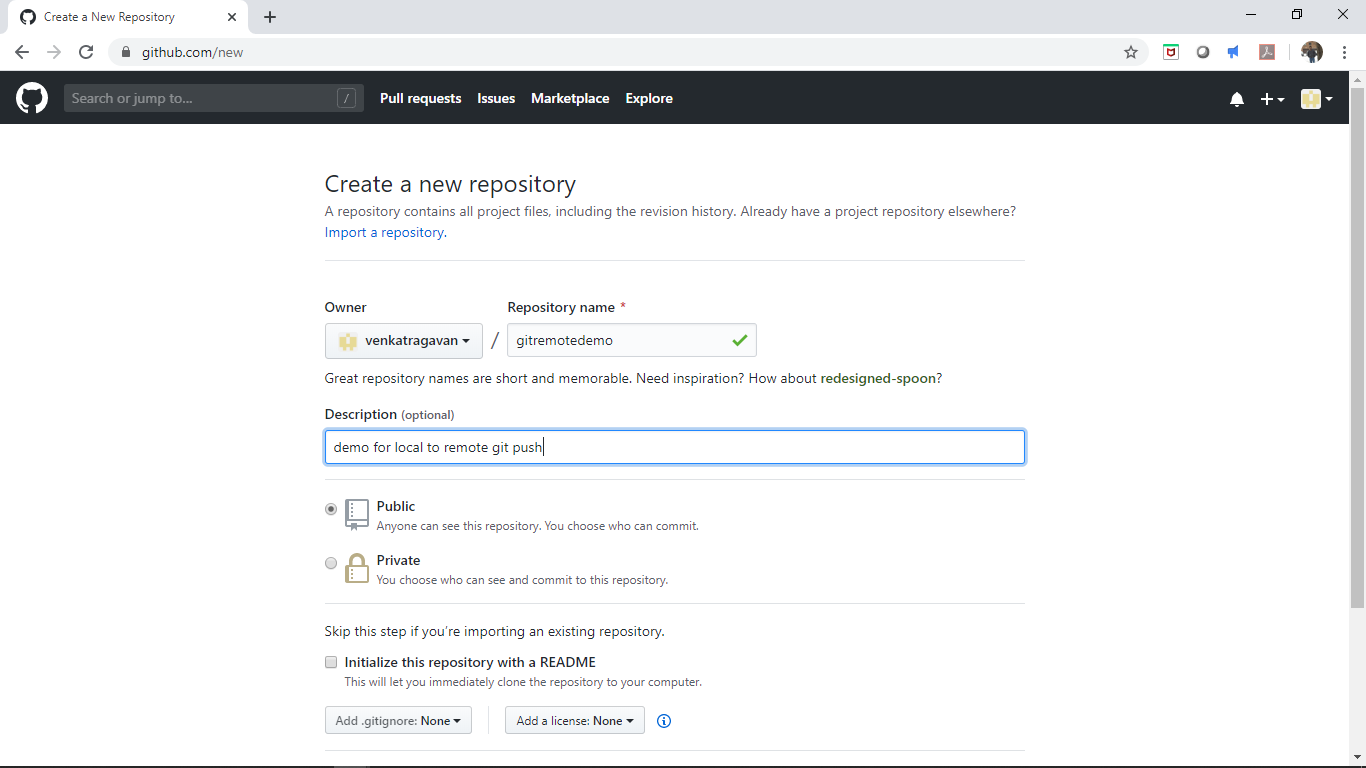


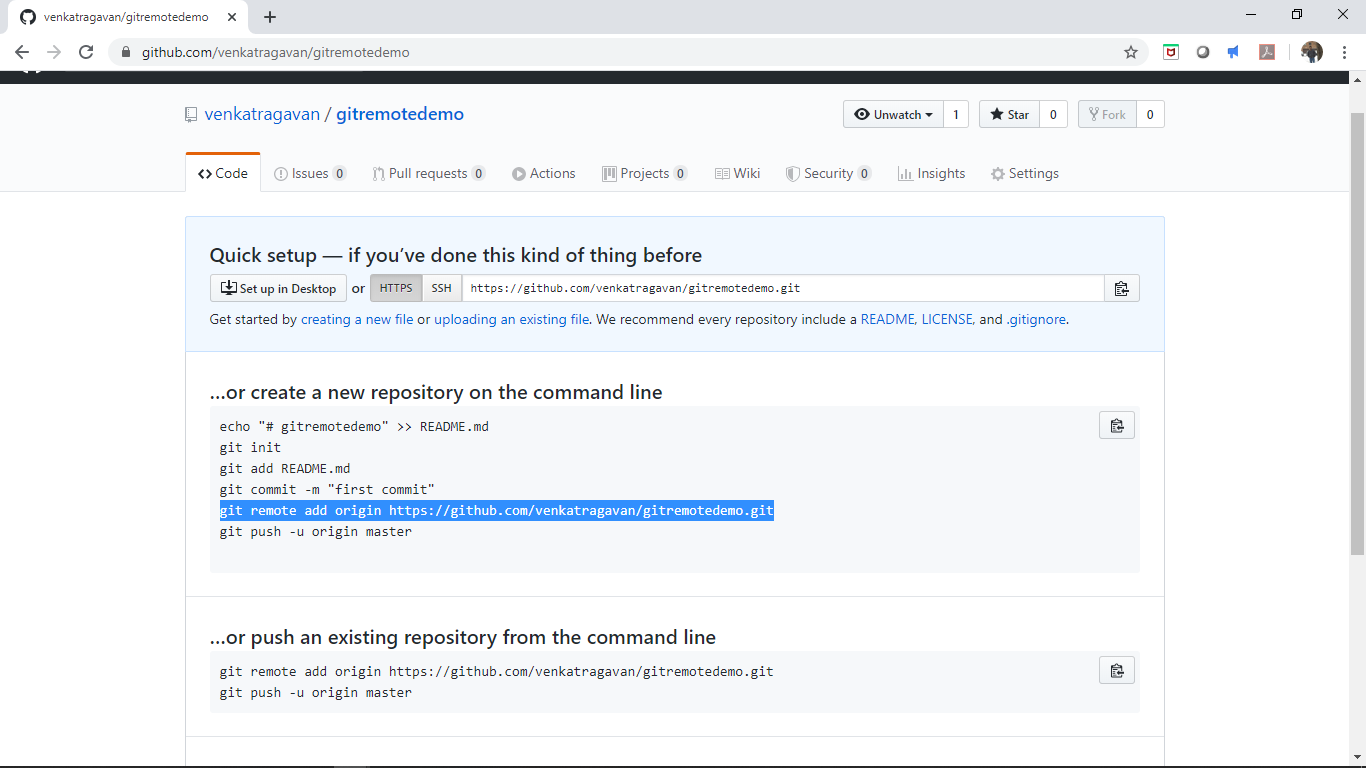


Projects are added to the local git, see below:

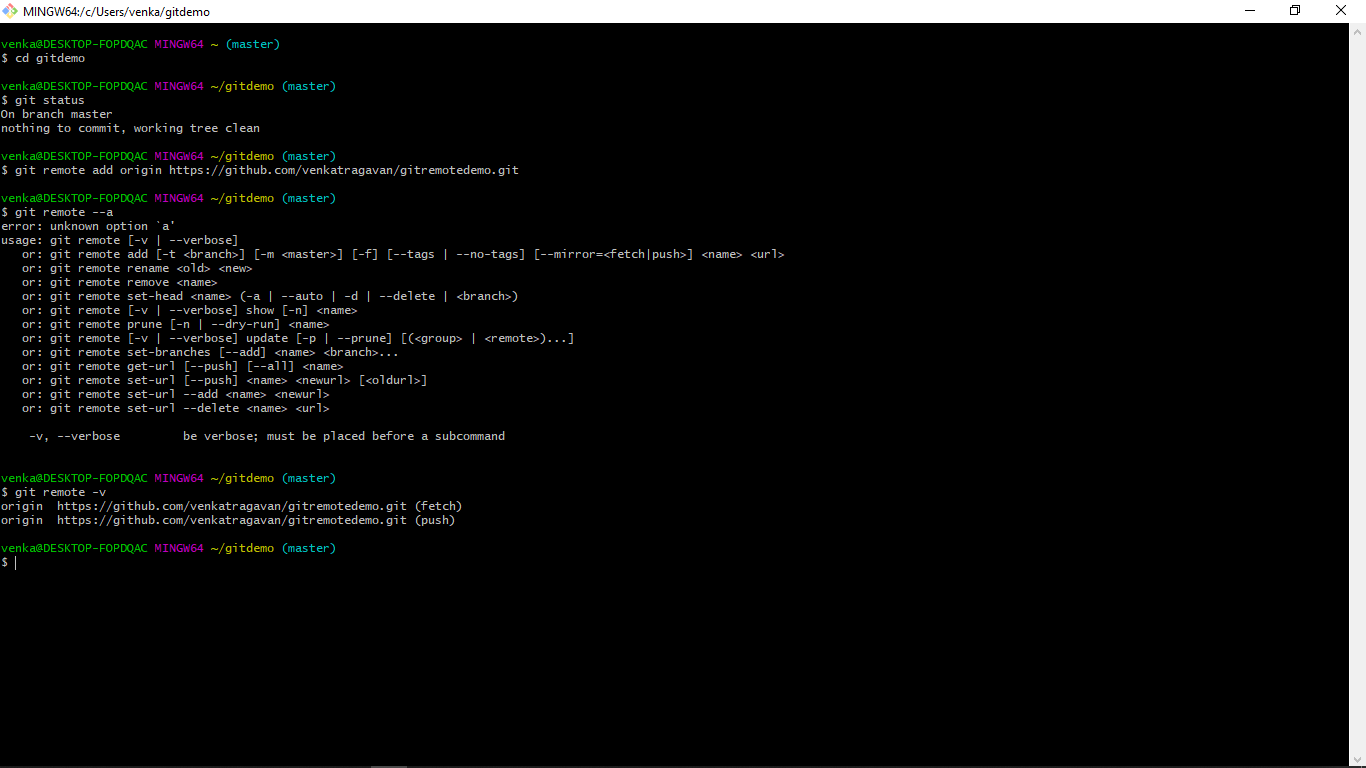


4. Push the project in assignment 2 to remote repository:

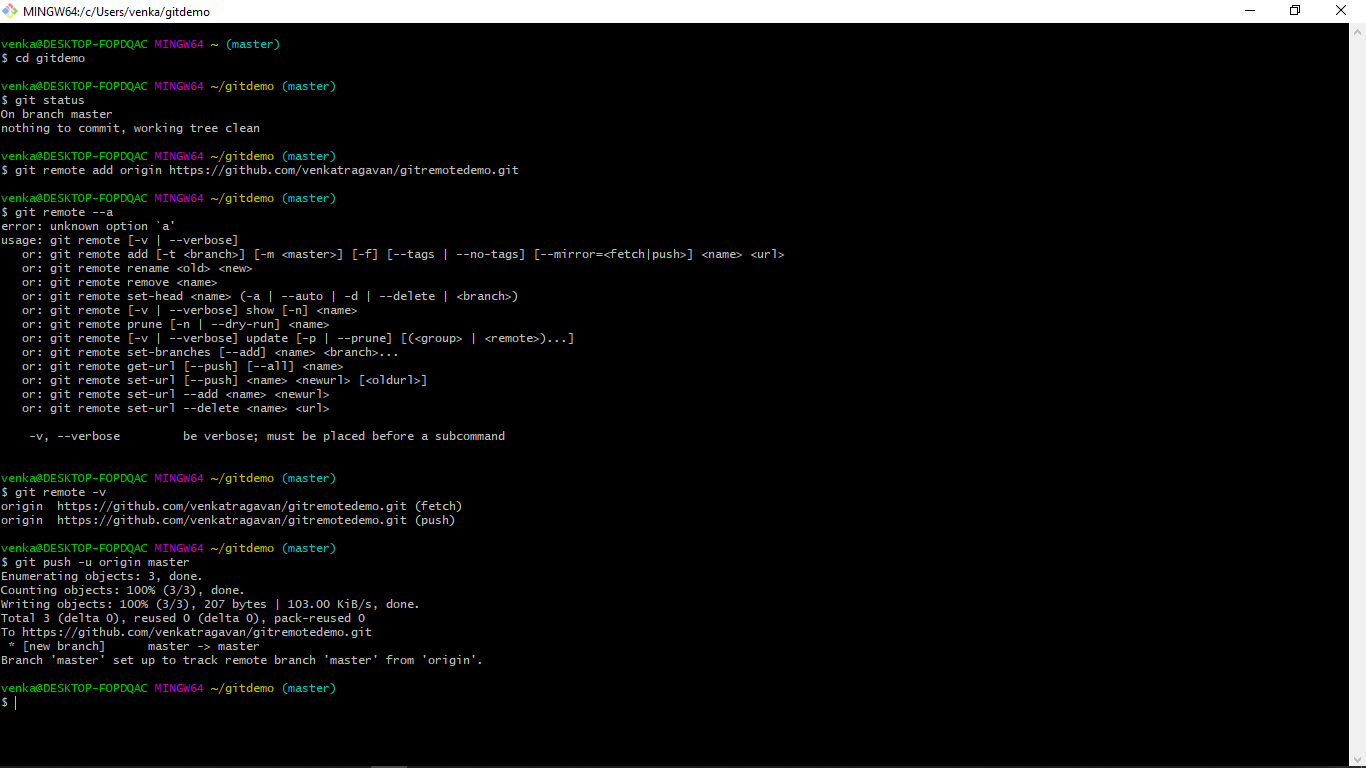


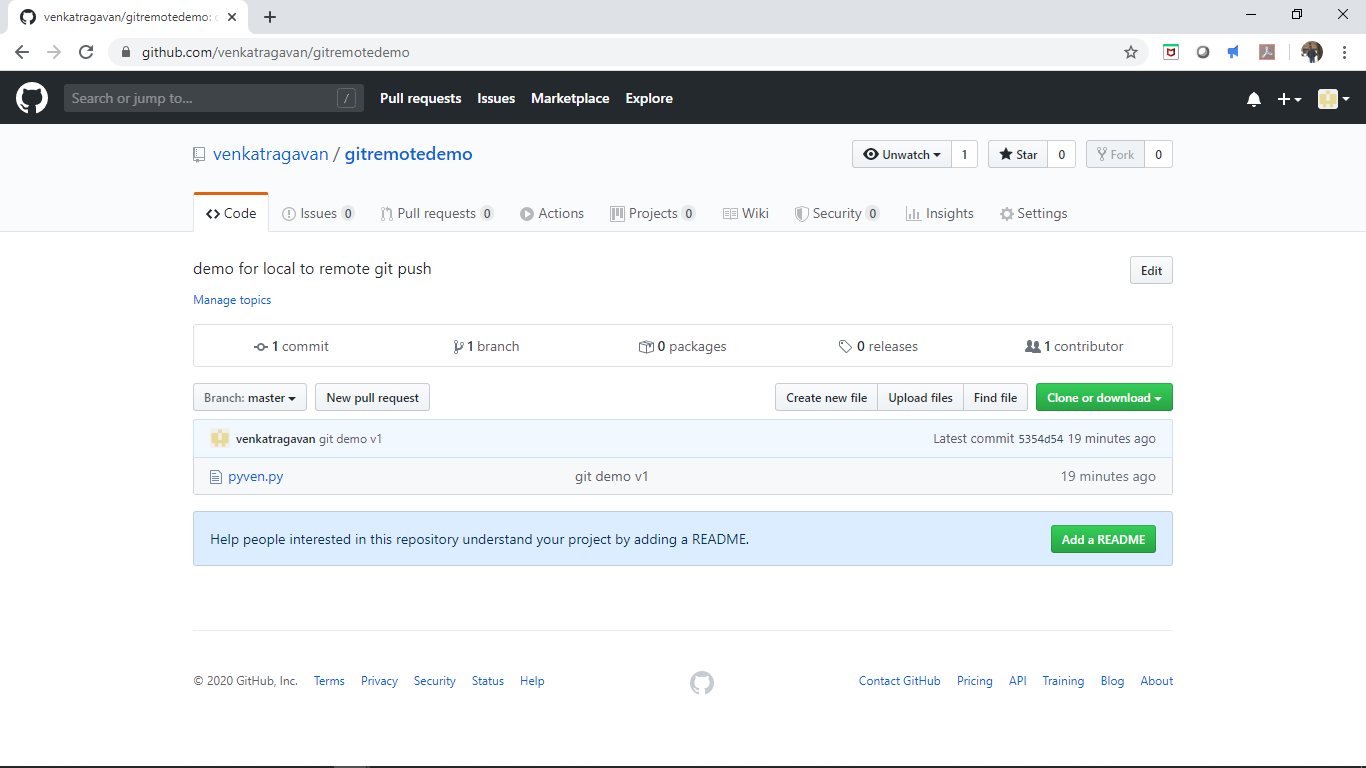


Added the remote repo url in the local git:



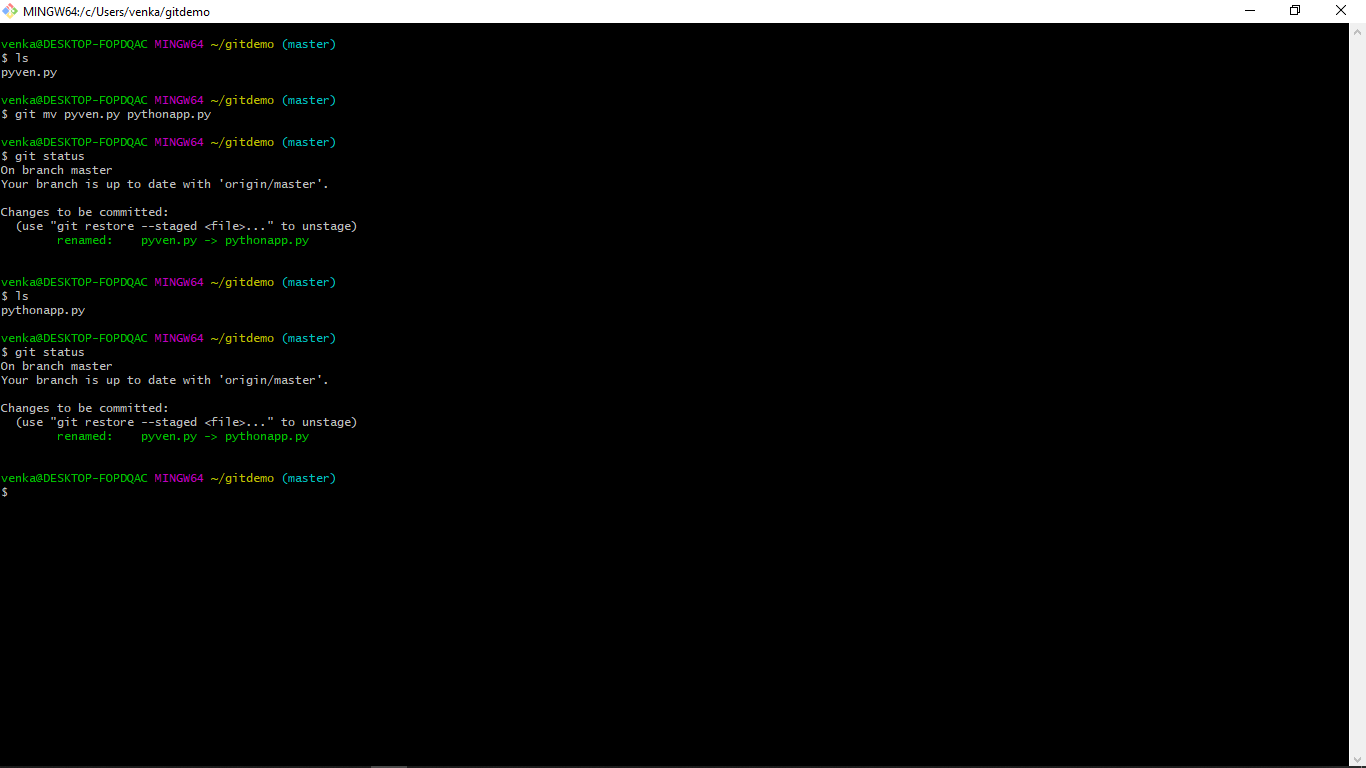
Pushed the changes in local repo to the remote origin master repo





5. Git move and renaming files:

Renaming the local repo file that is already committed as shown below:

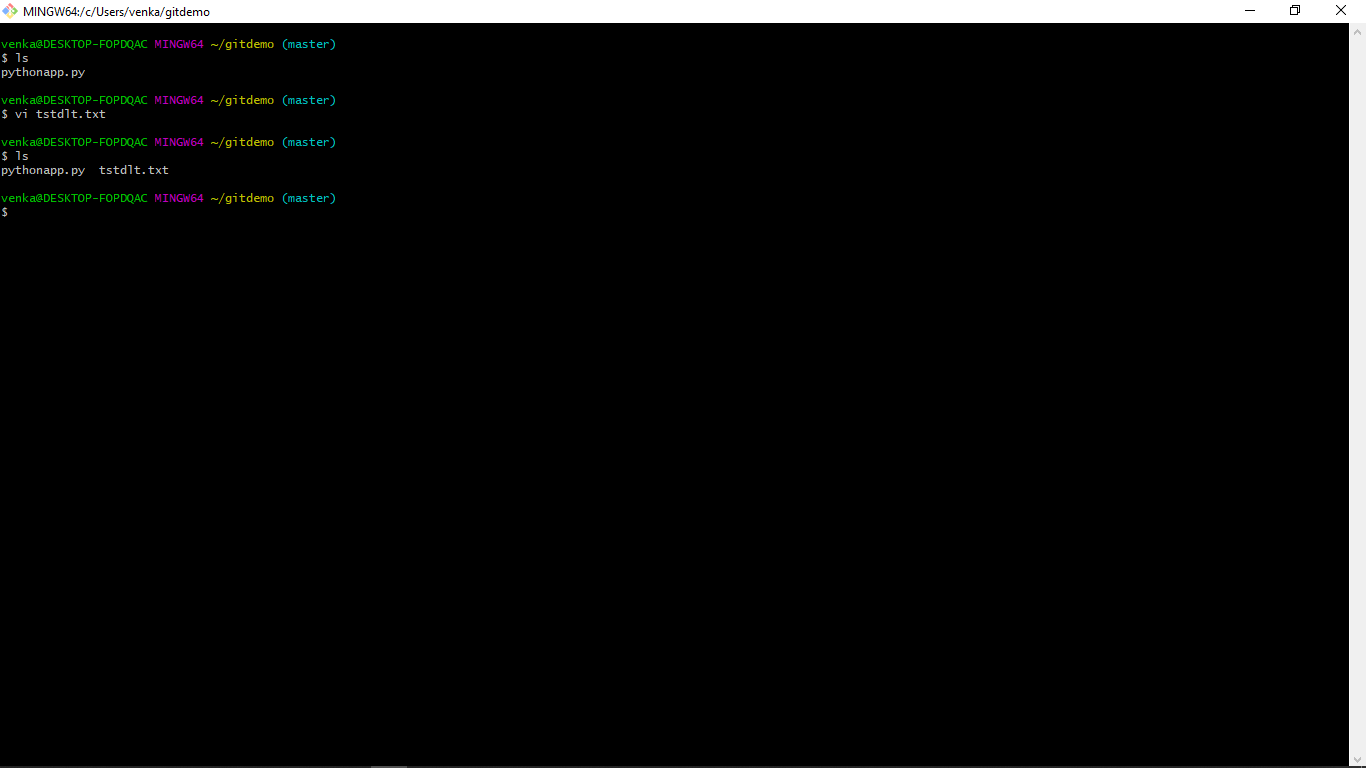


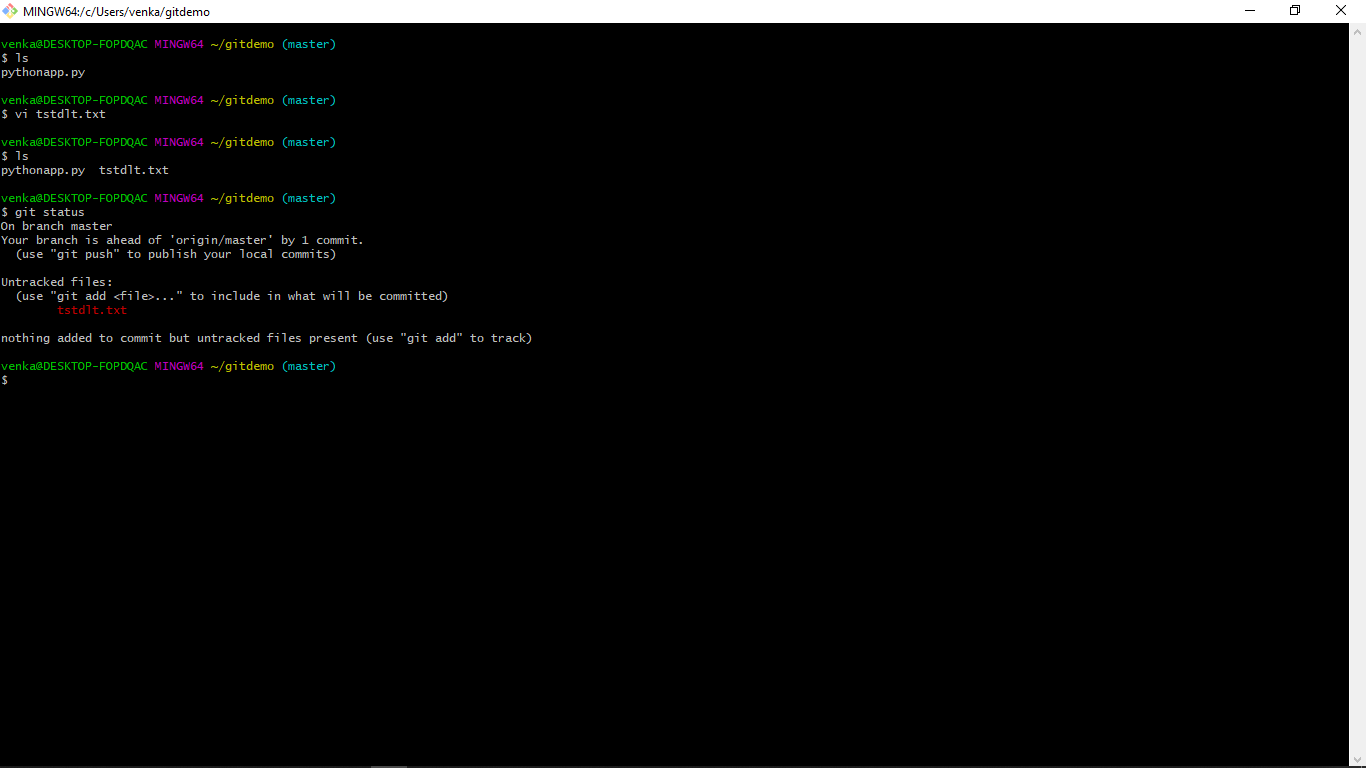
Renaming is committed:



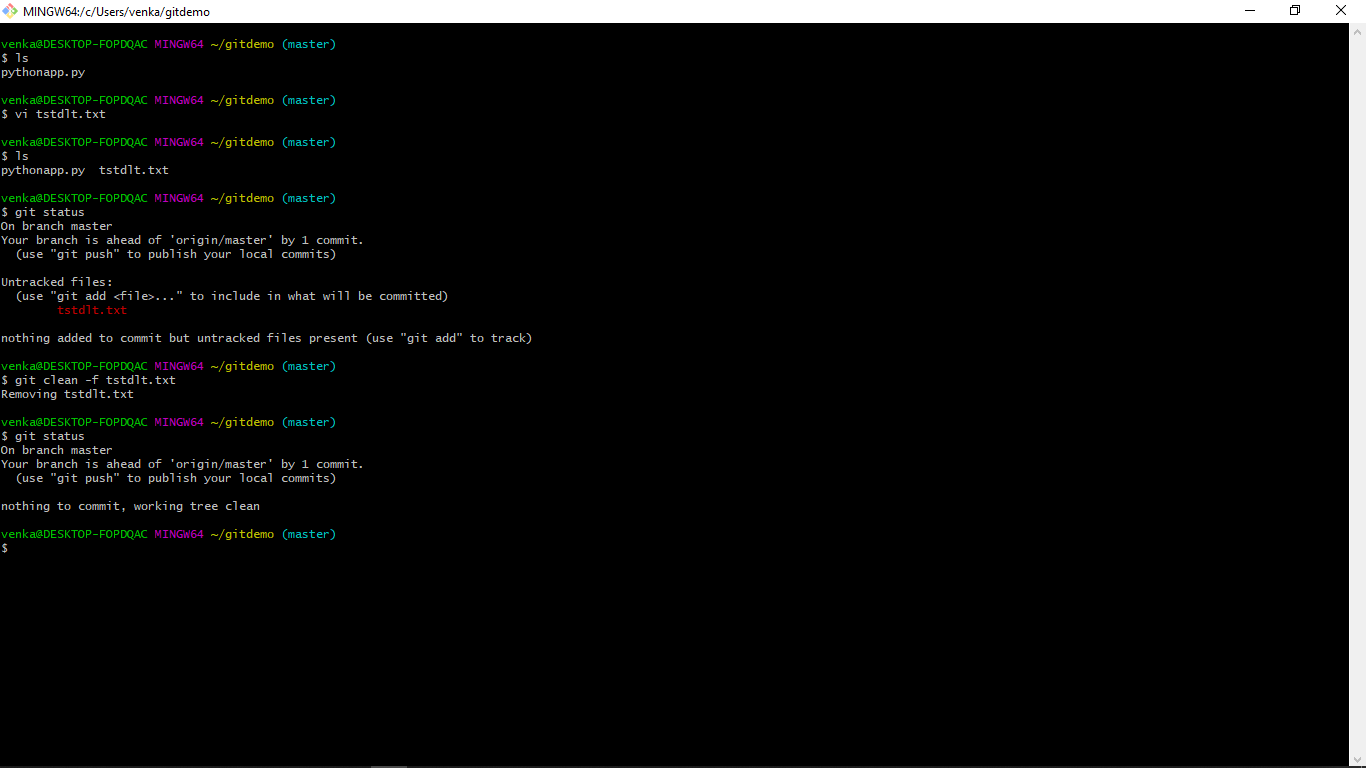
6. Deleting the untracked file:

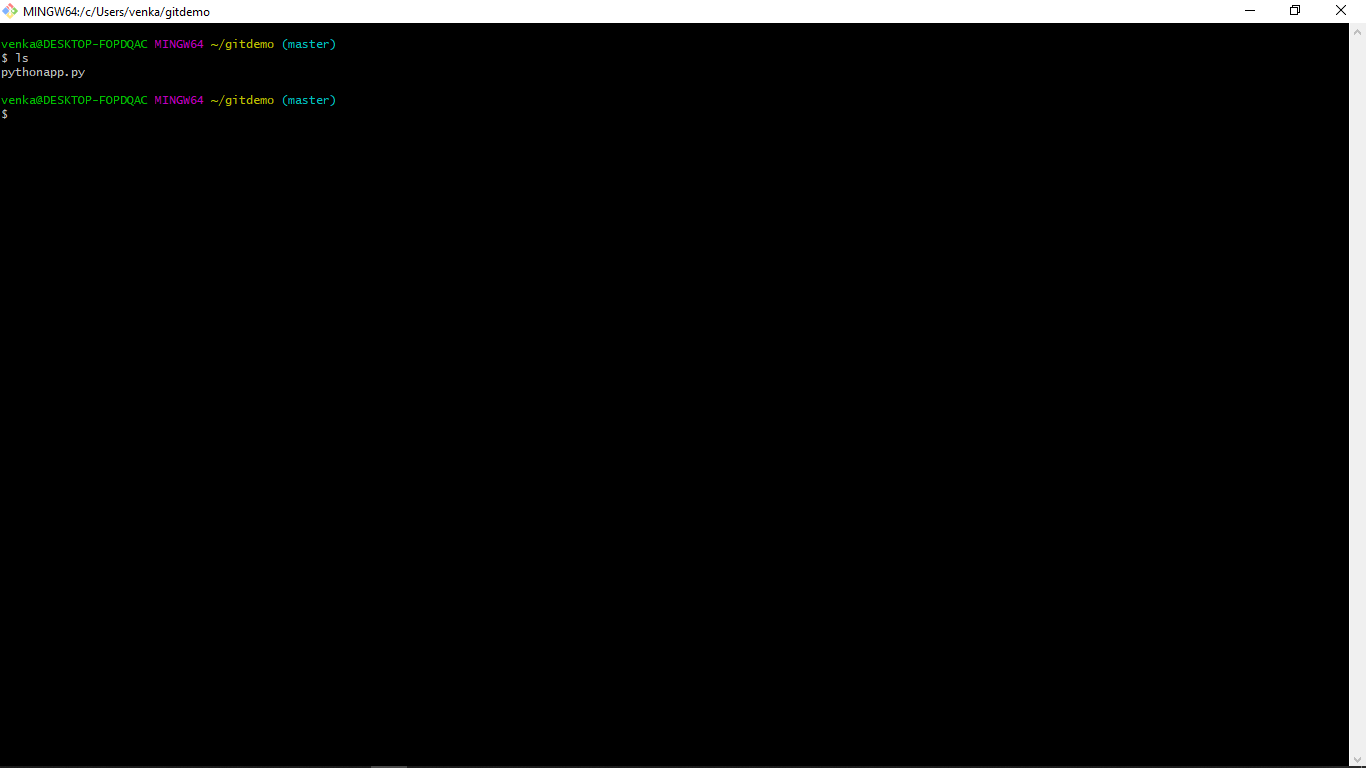
Created a new file in the existing git folder





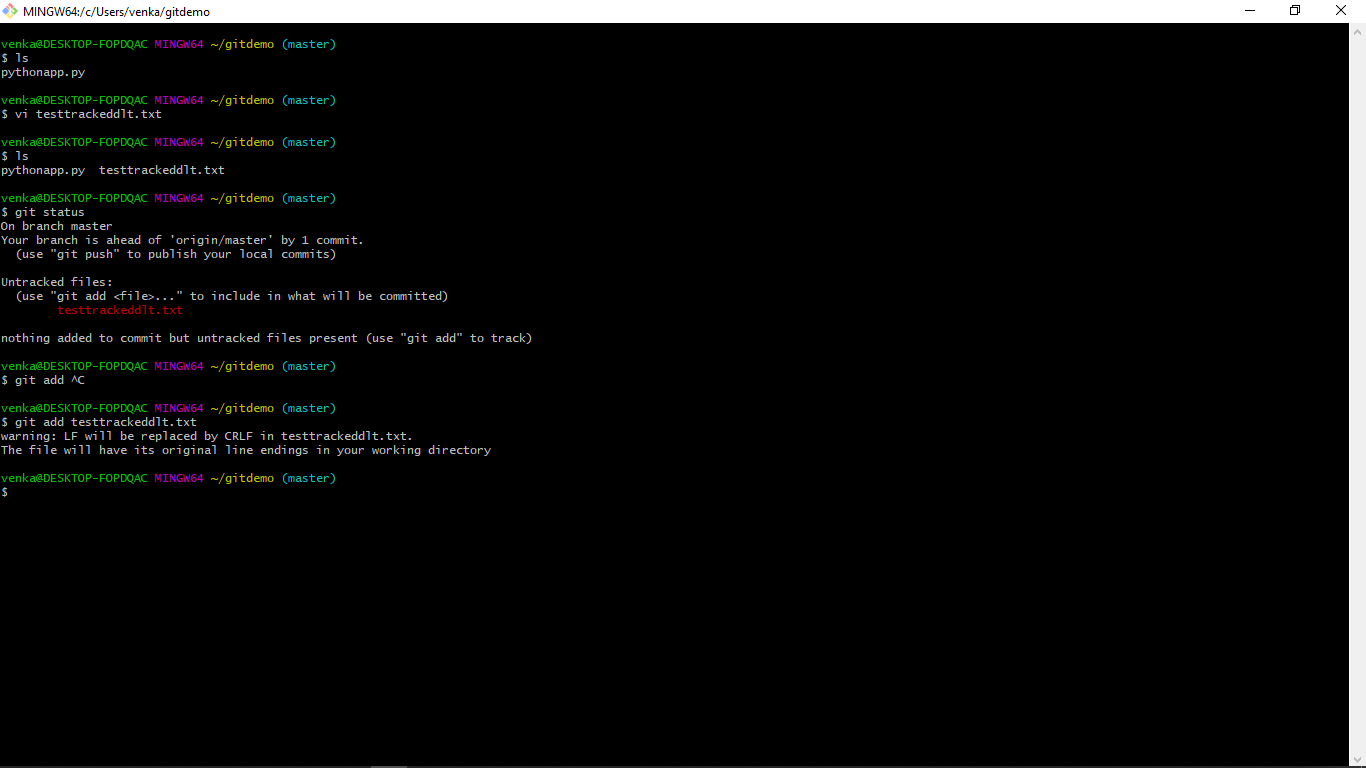
Did a git clean -f tstdlt.txt, please see below:

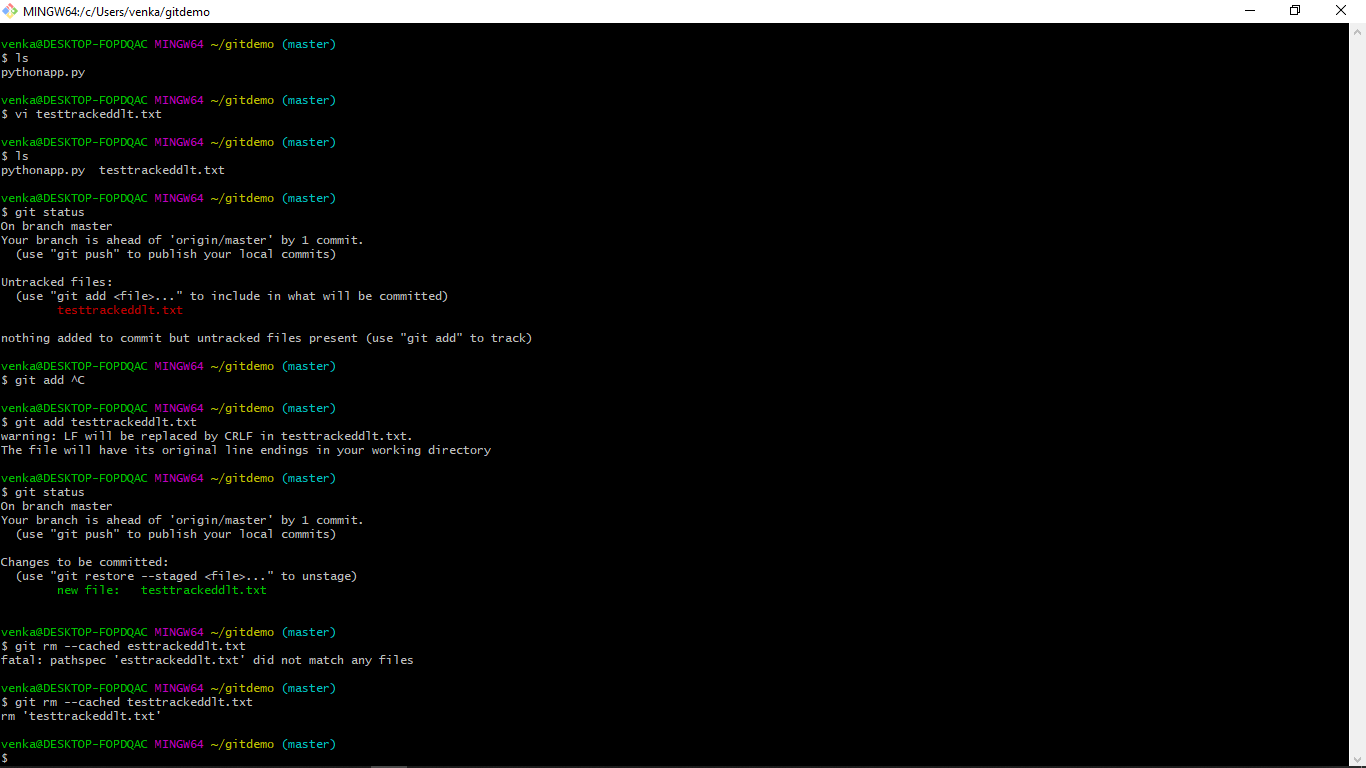




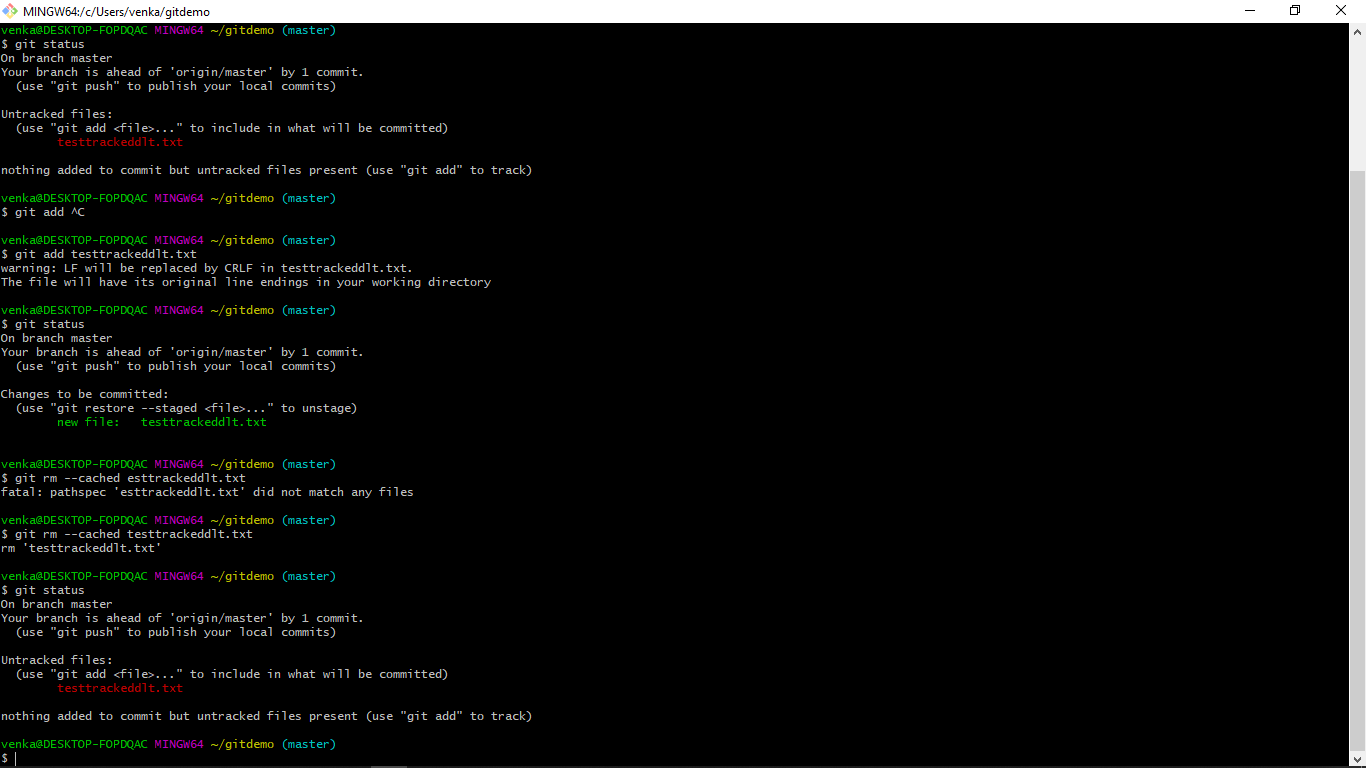
7 a) Delete a tracked file:

Created a new file and added it to staging directory where the file is tracked:

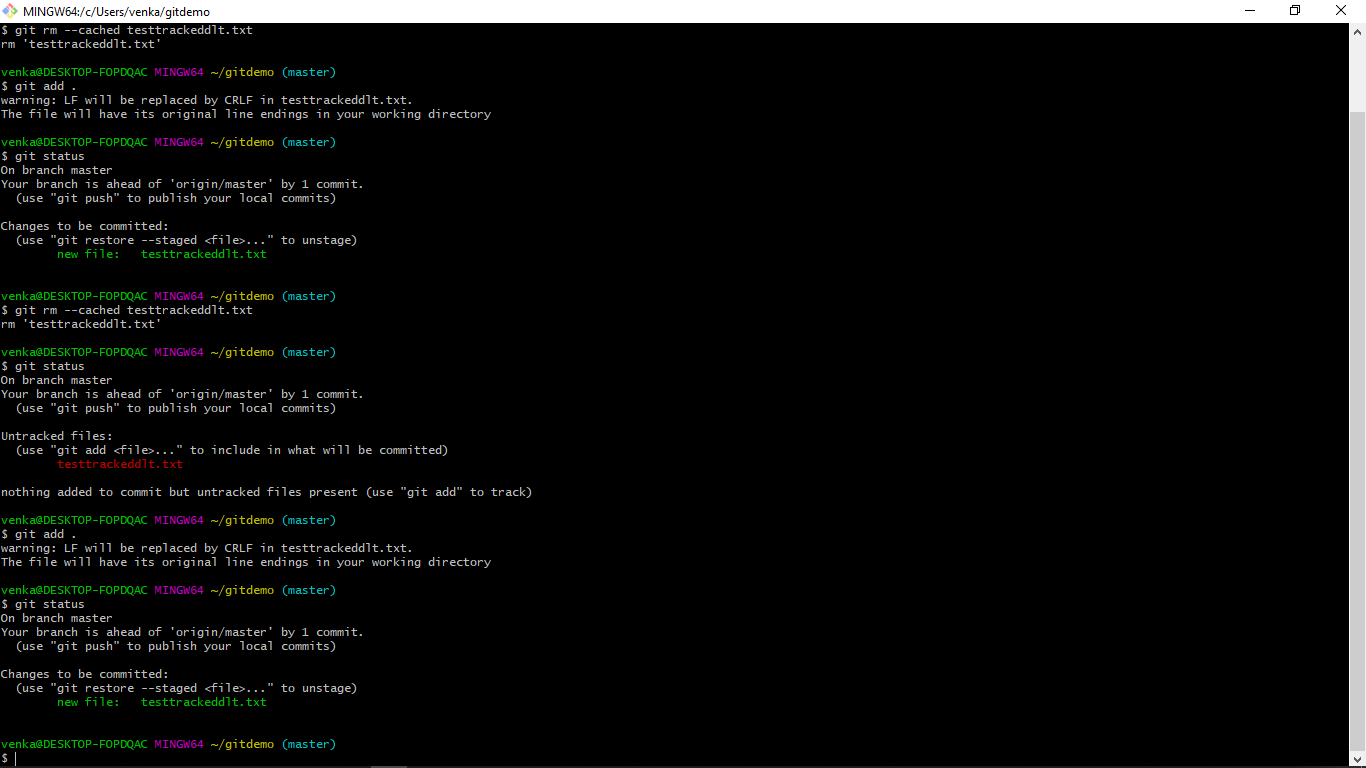




Now the file is untracked:



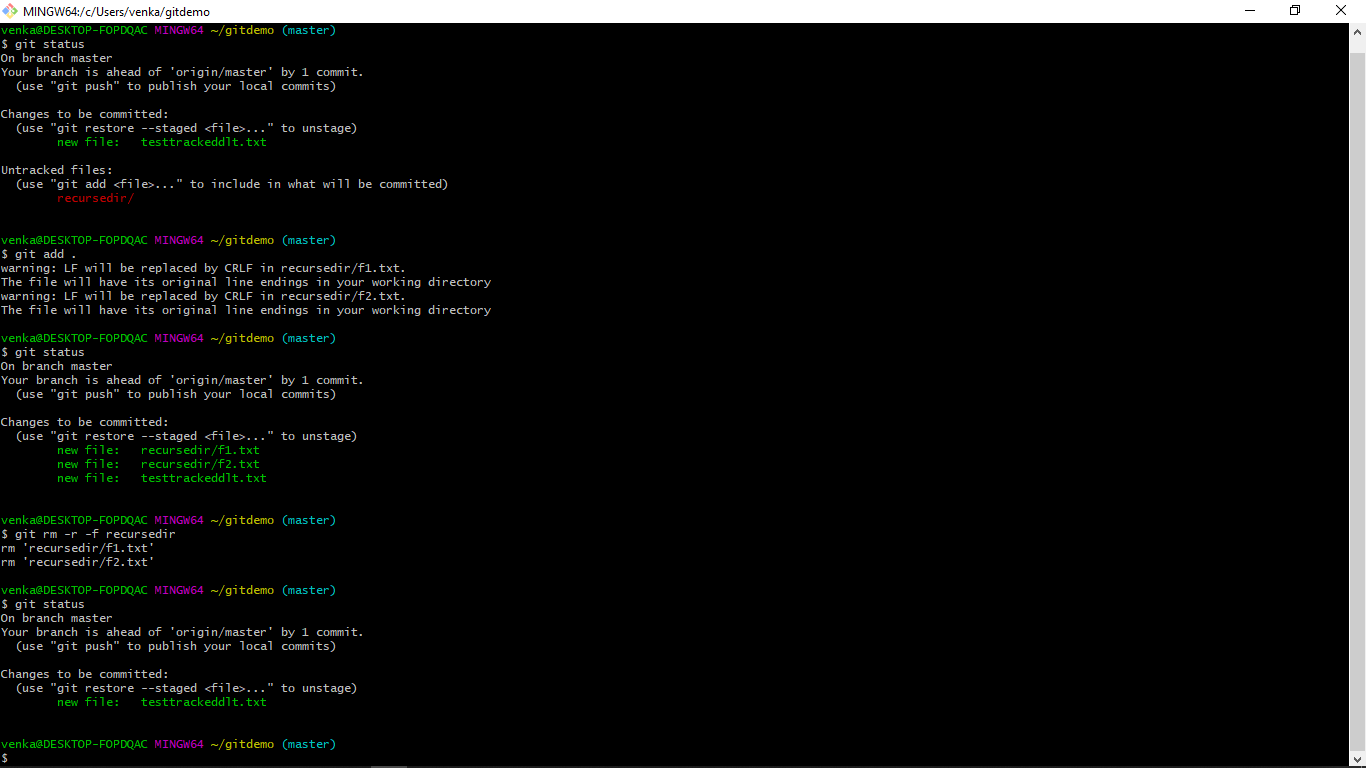
7B. Backing out staged deletion



7C. Recursive deletion

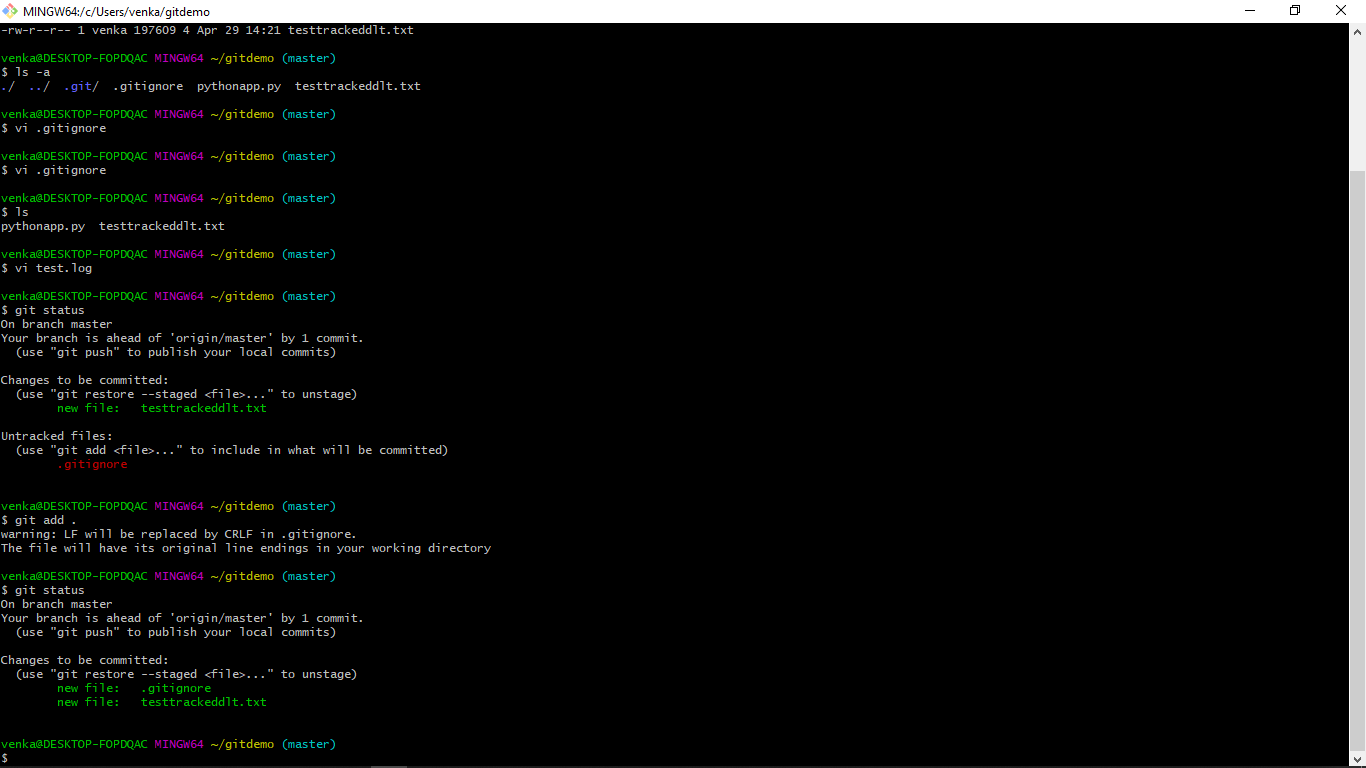
Created a directory and add two files inside that folder. Added that to stage and did a

recursive delete:

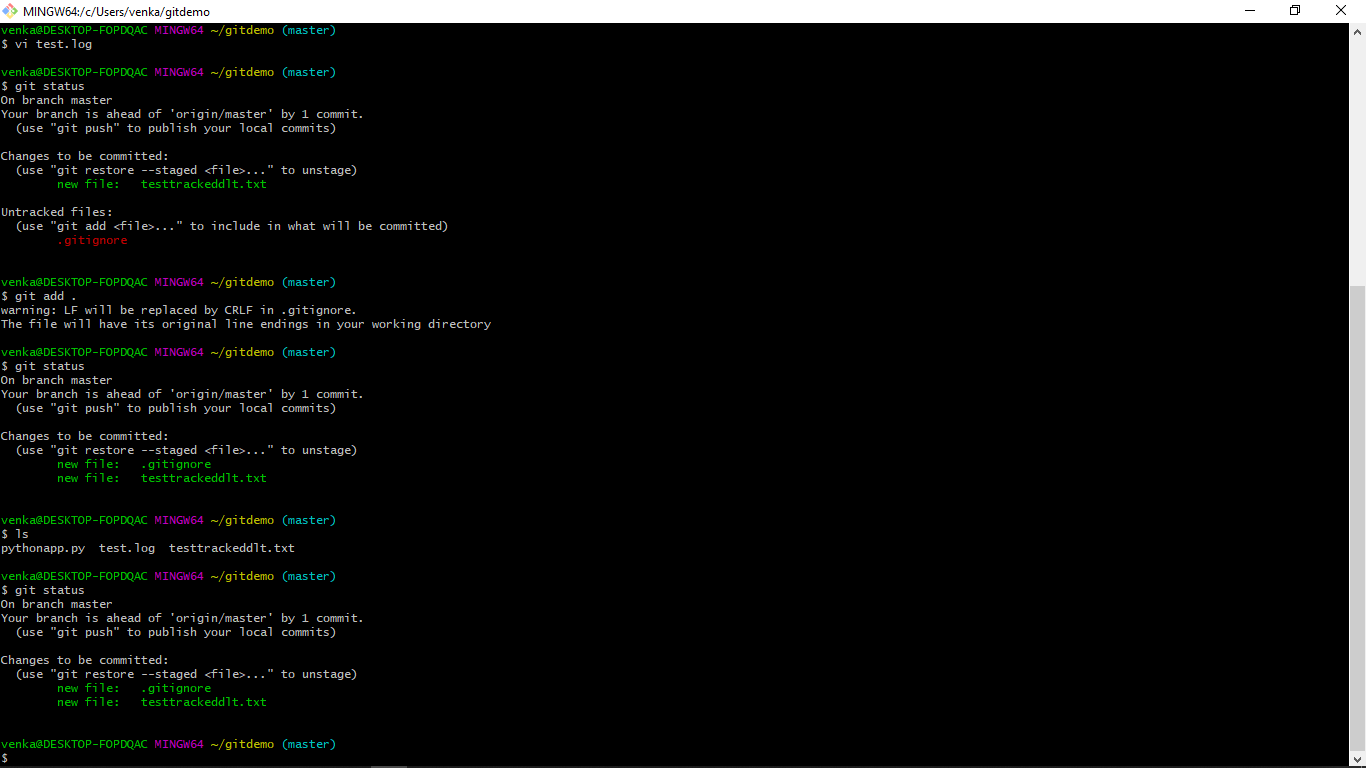


8. Ignoring certain files from being pushed into git.

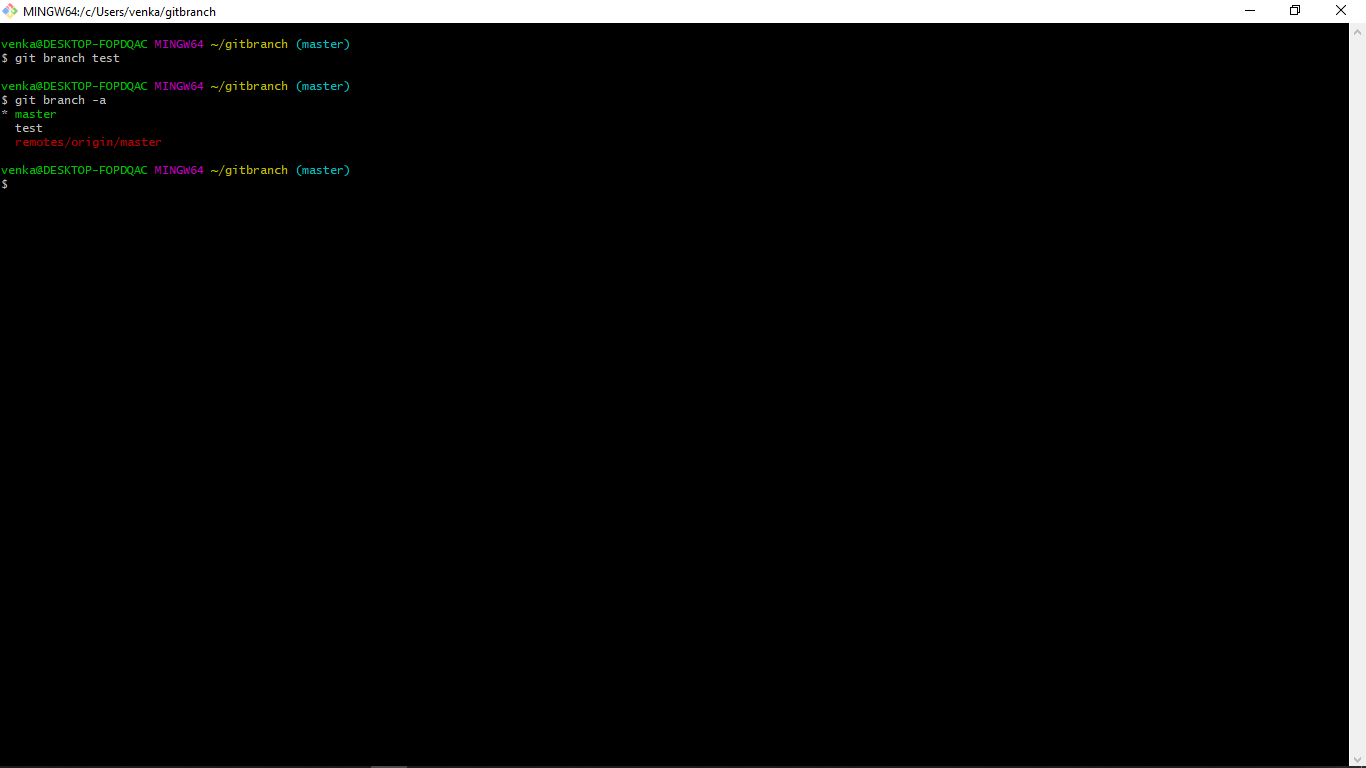


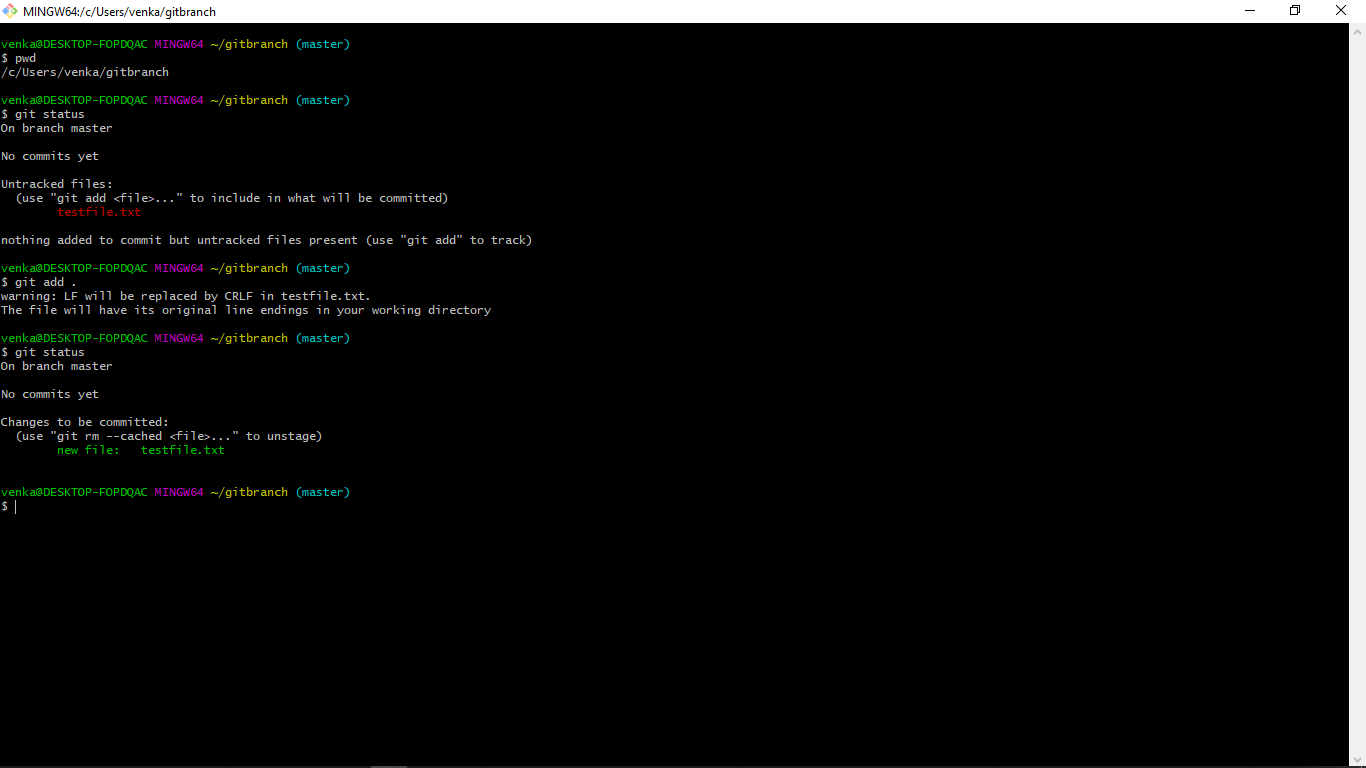


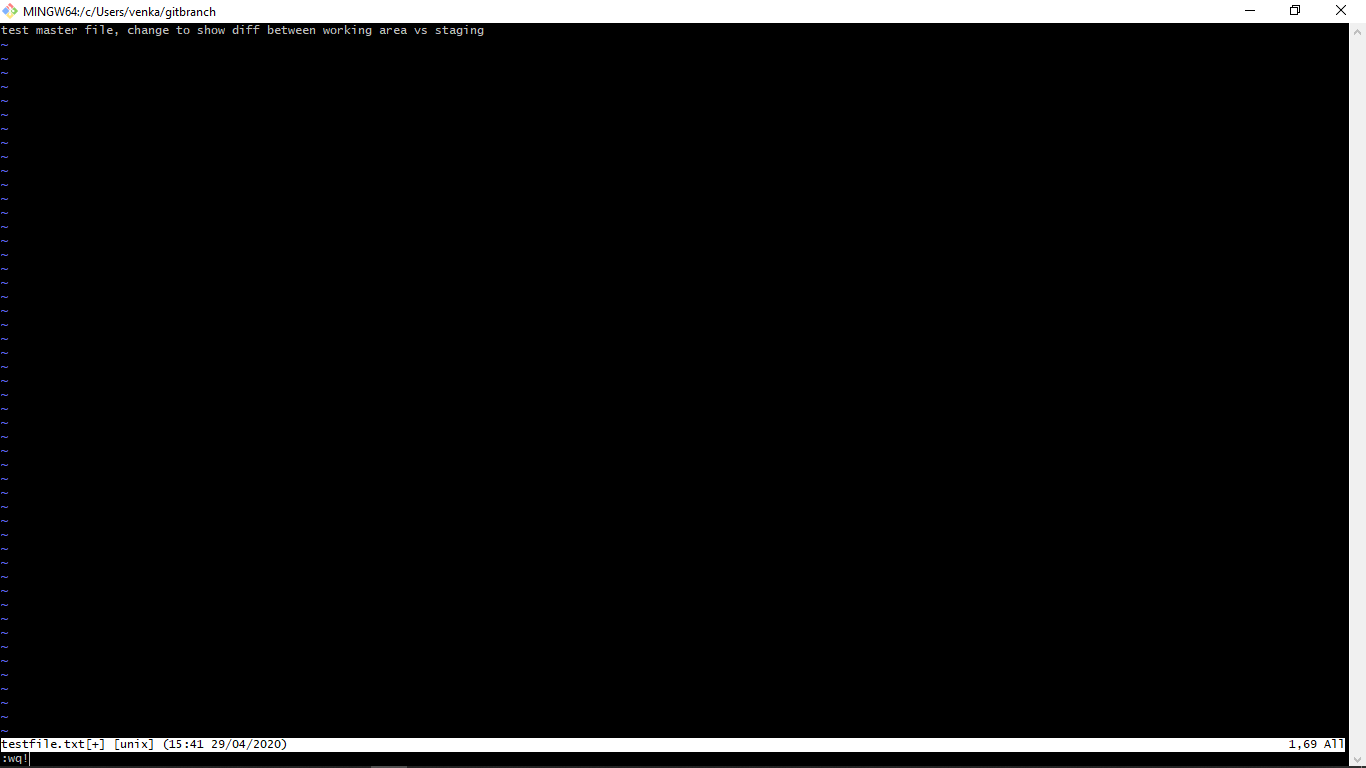
We can see the test.log in the working directory is not considered while staging as the .log entry is ignored under .gitignore file.



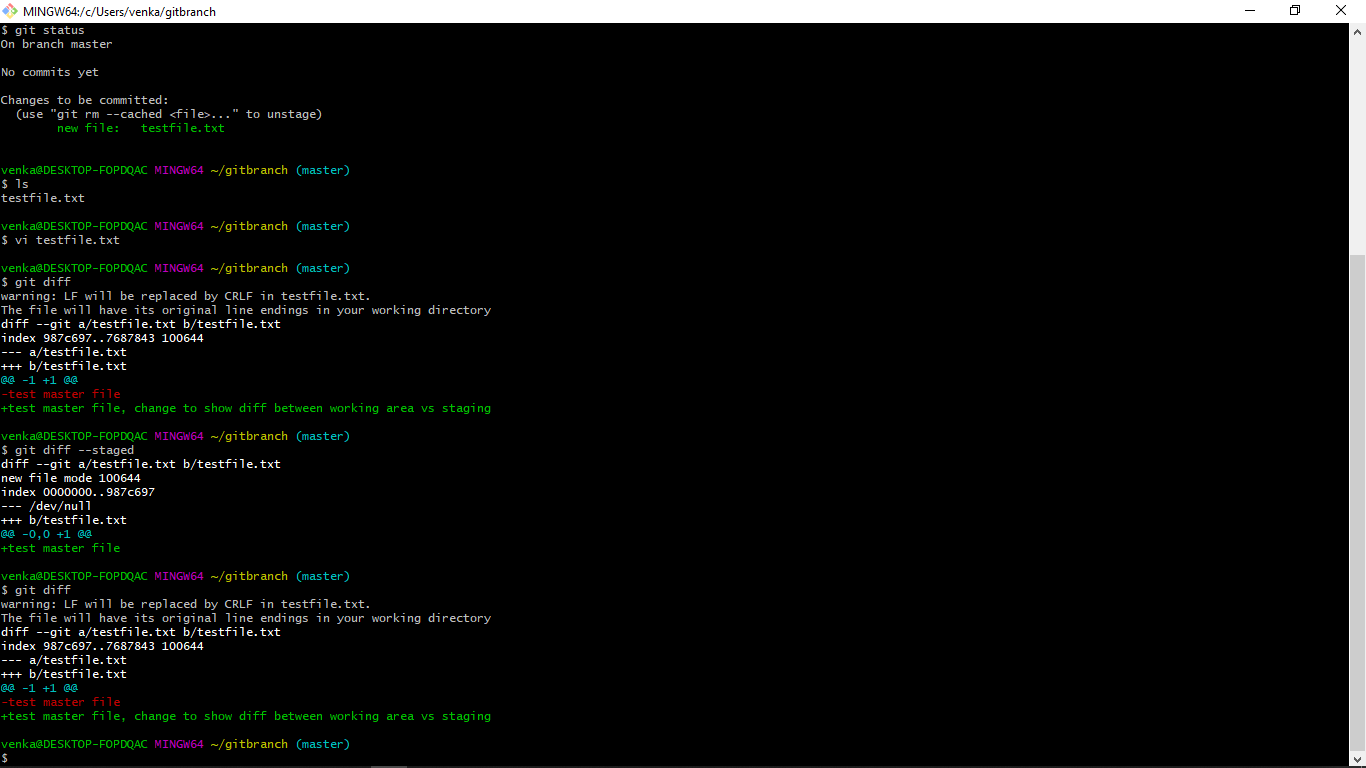
9. Create a new branch test:



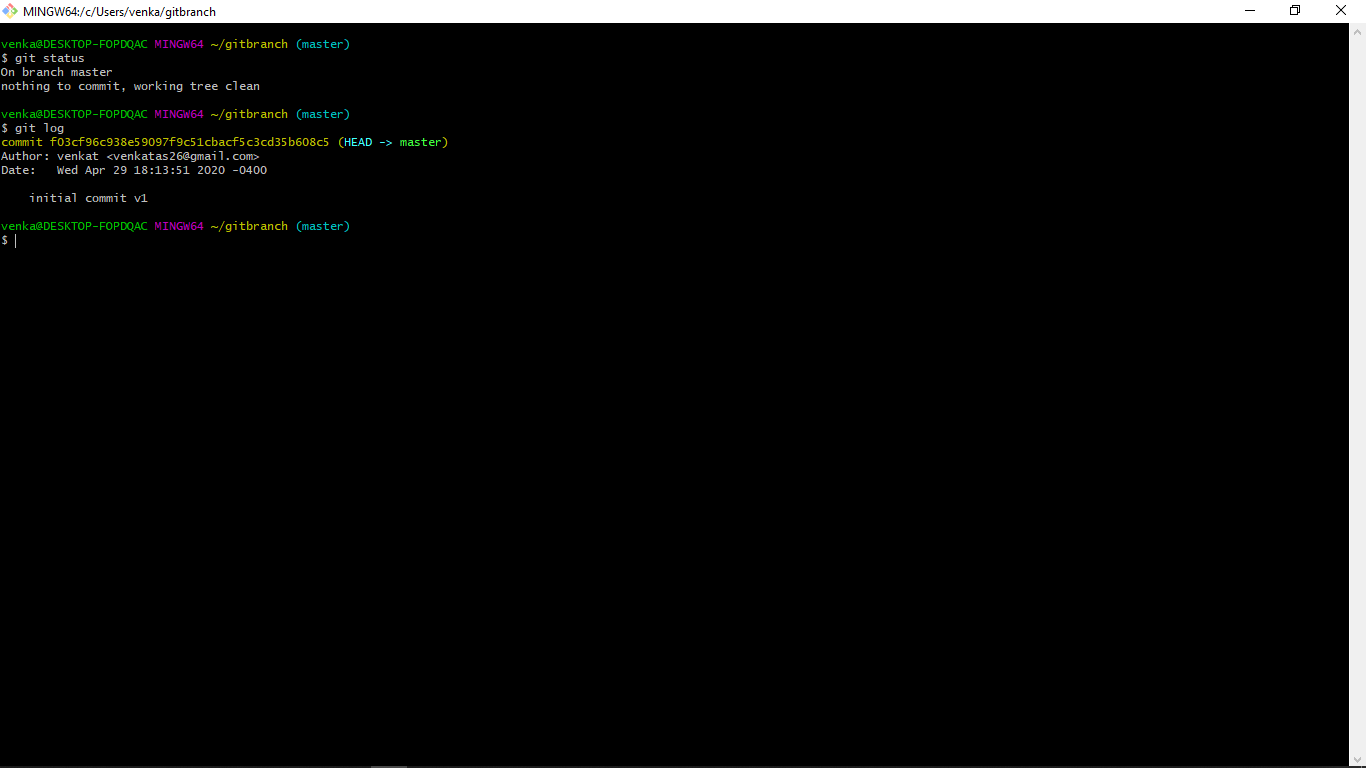




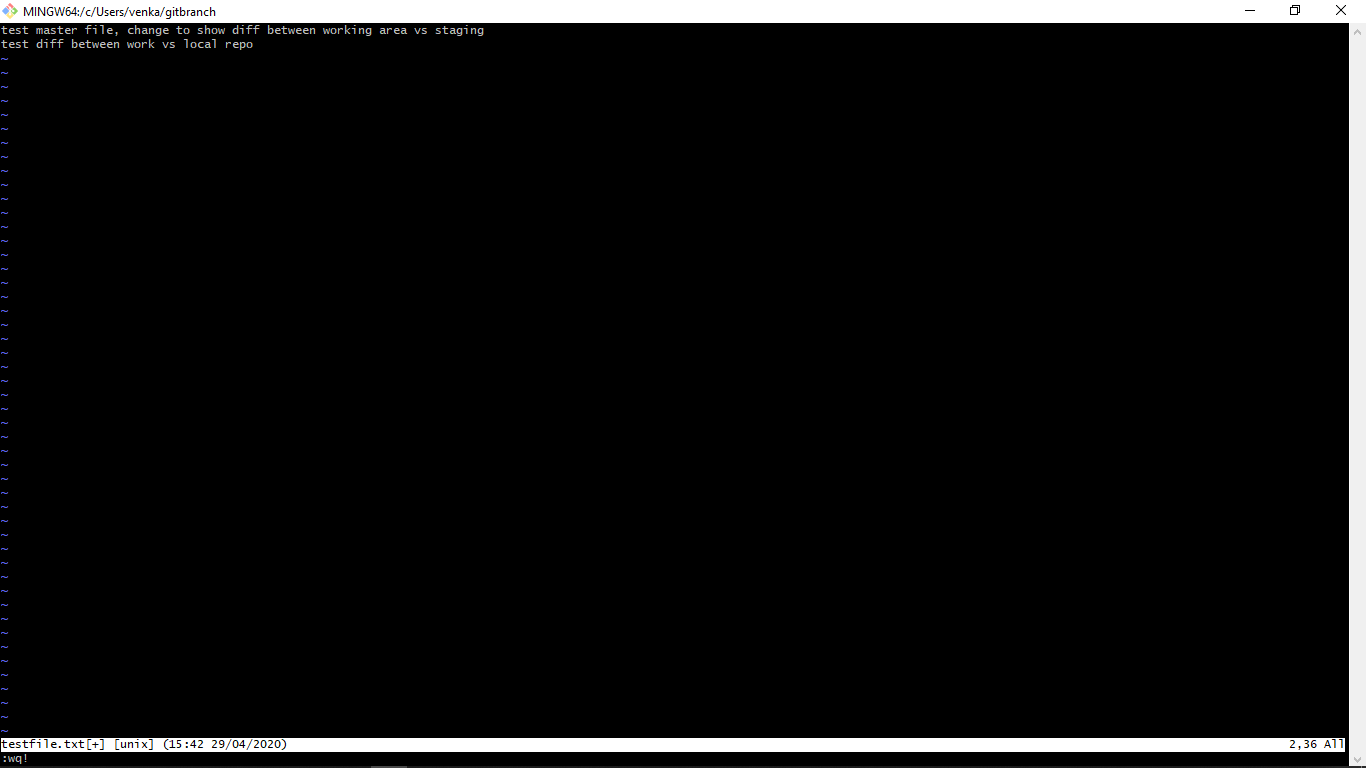
1. Working directory vs Staging area



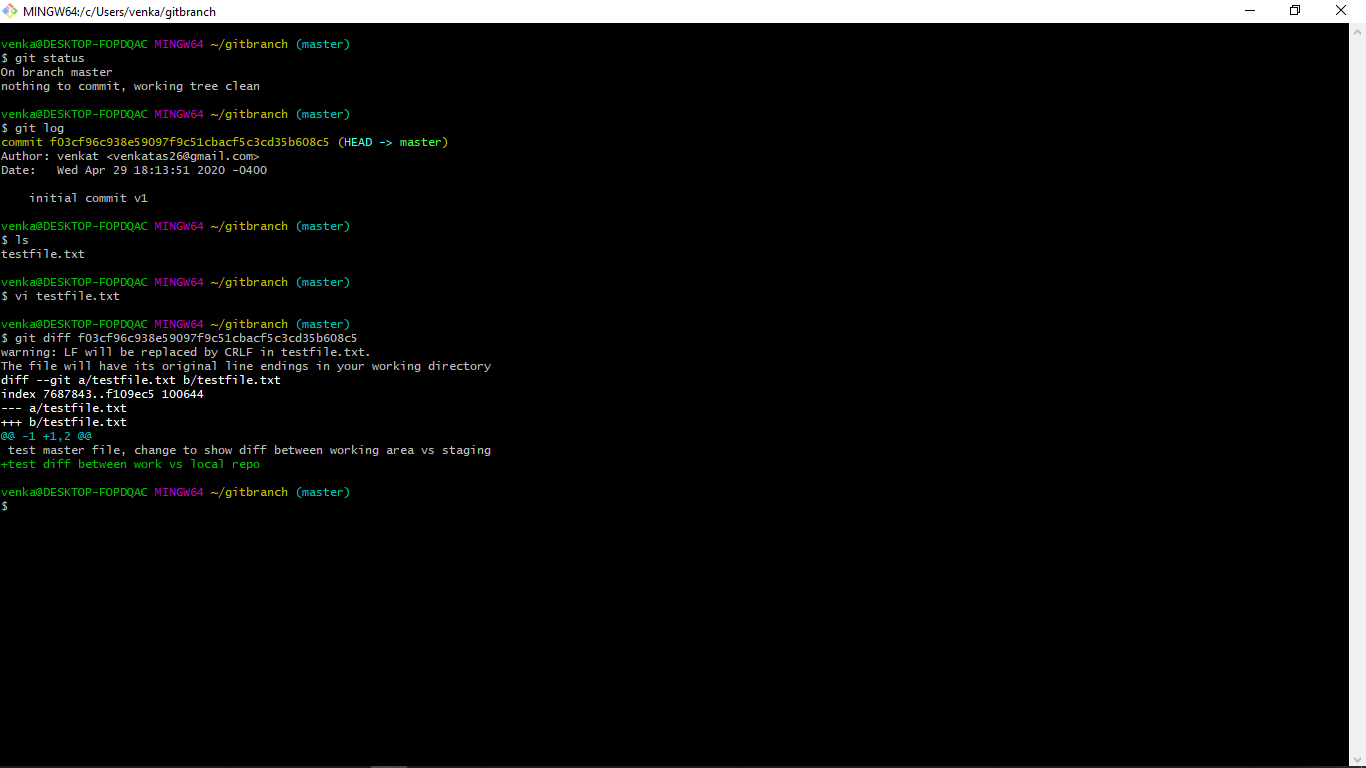
Did a commit to local repo:



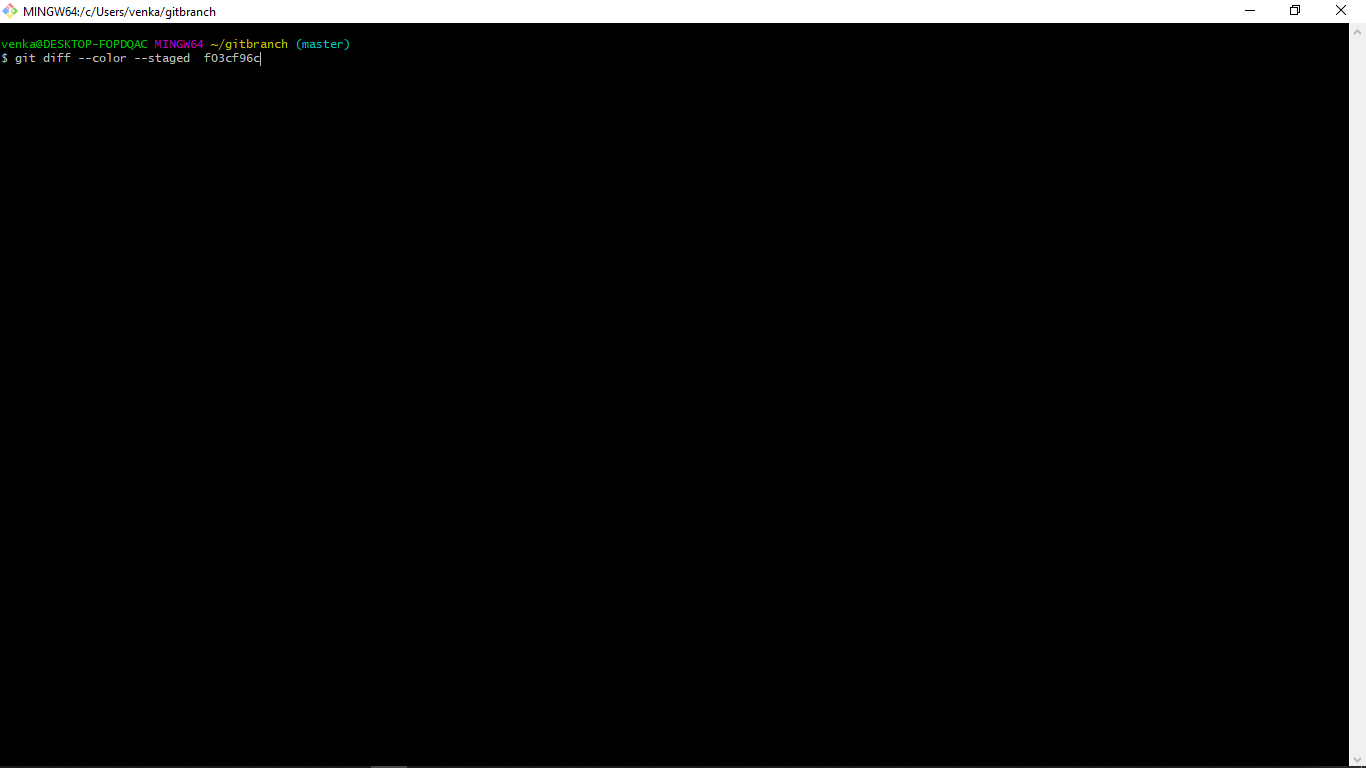
Changed the working directory file after commit to local repo

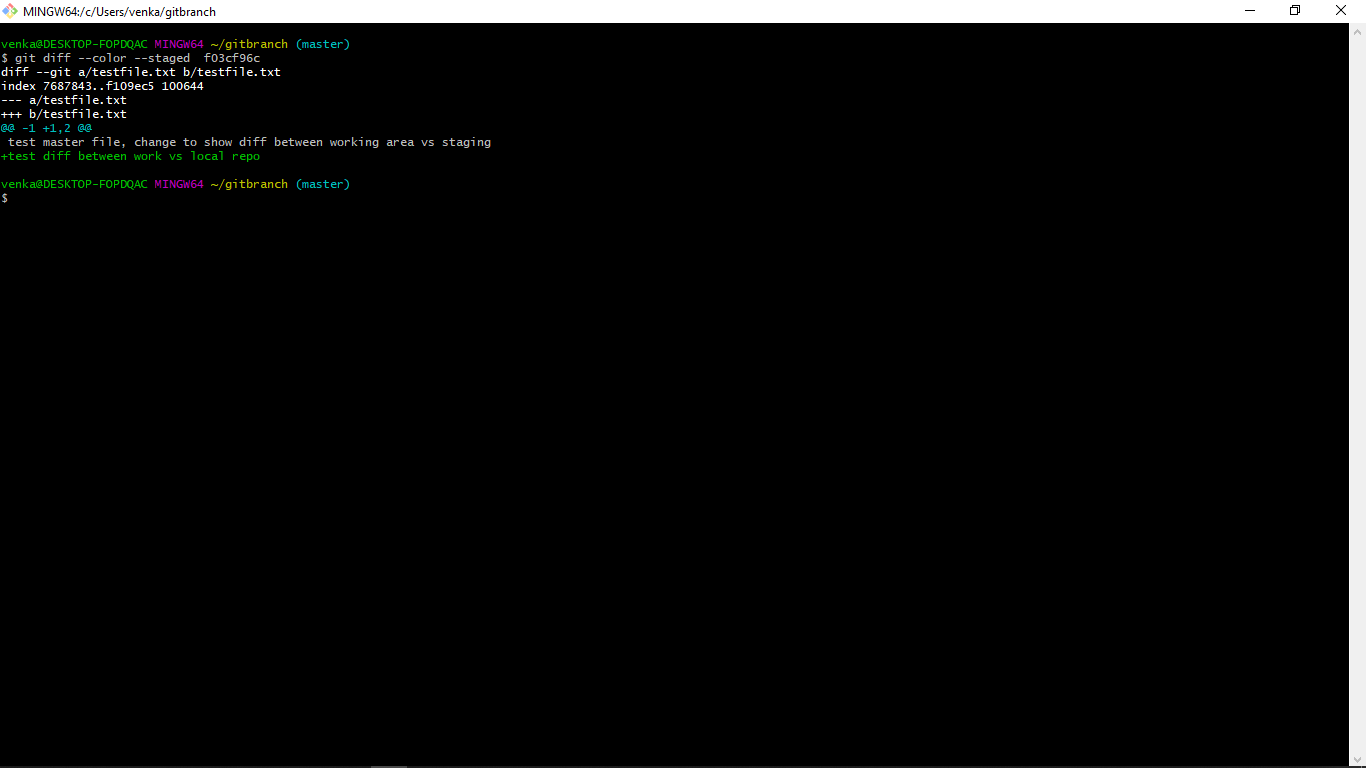


1. Working area vs Local repo after changing working directory:

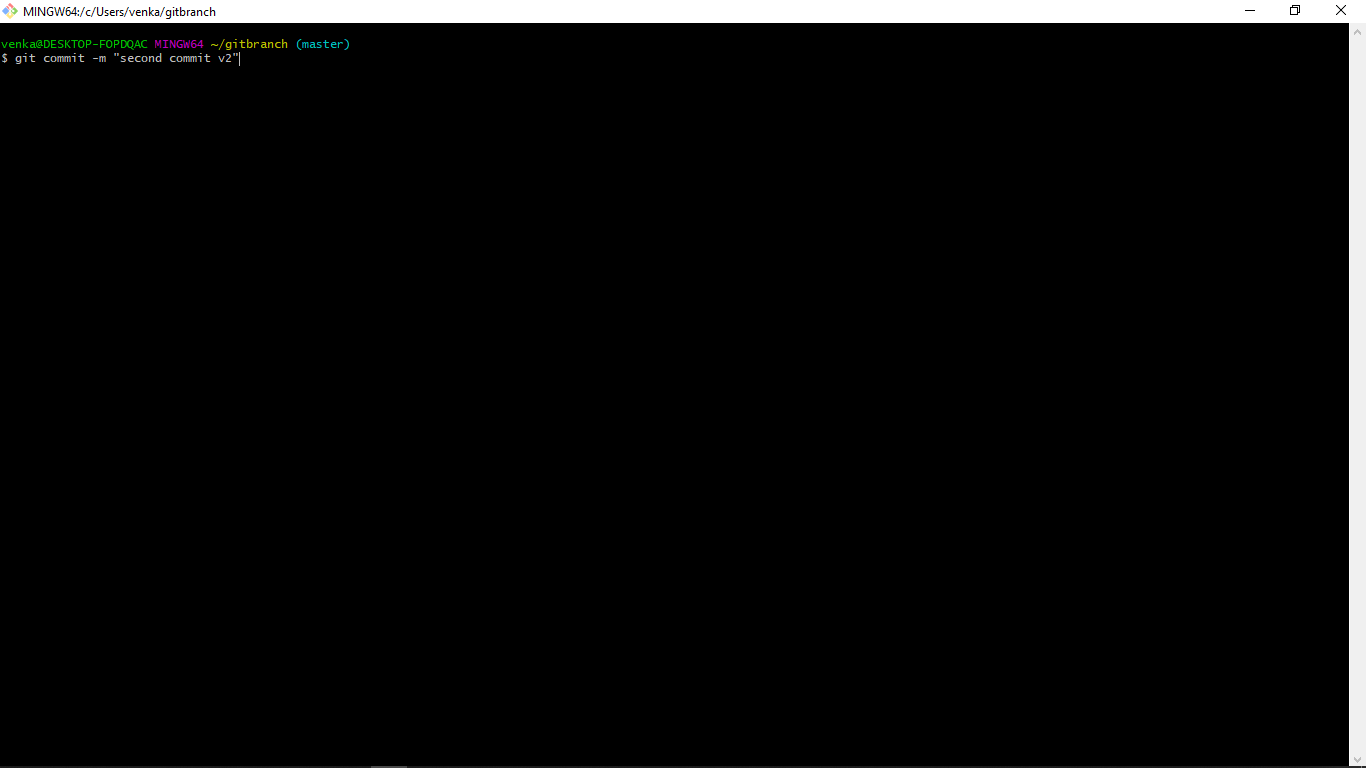


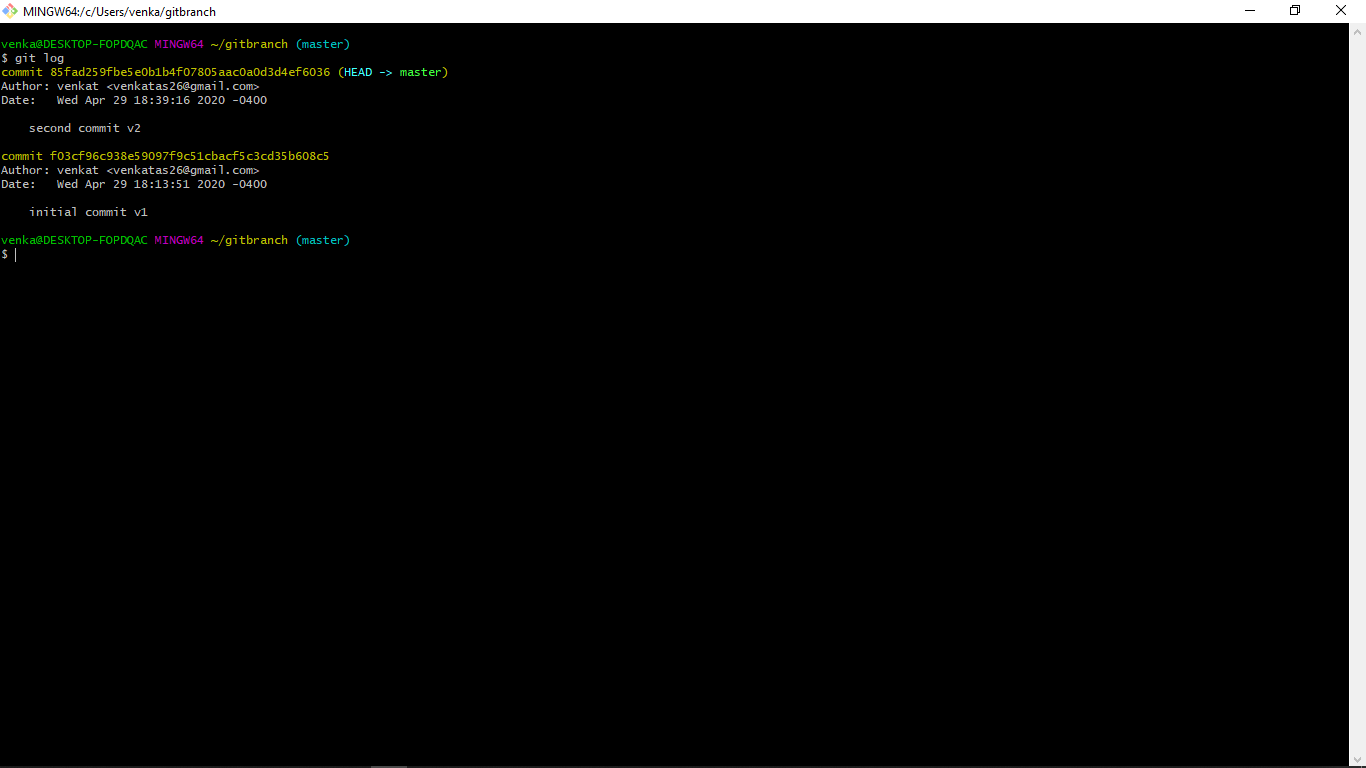
1. Staging area vs Local repository.



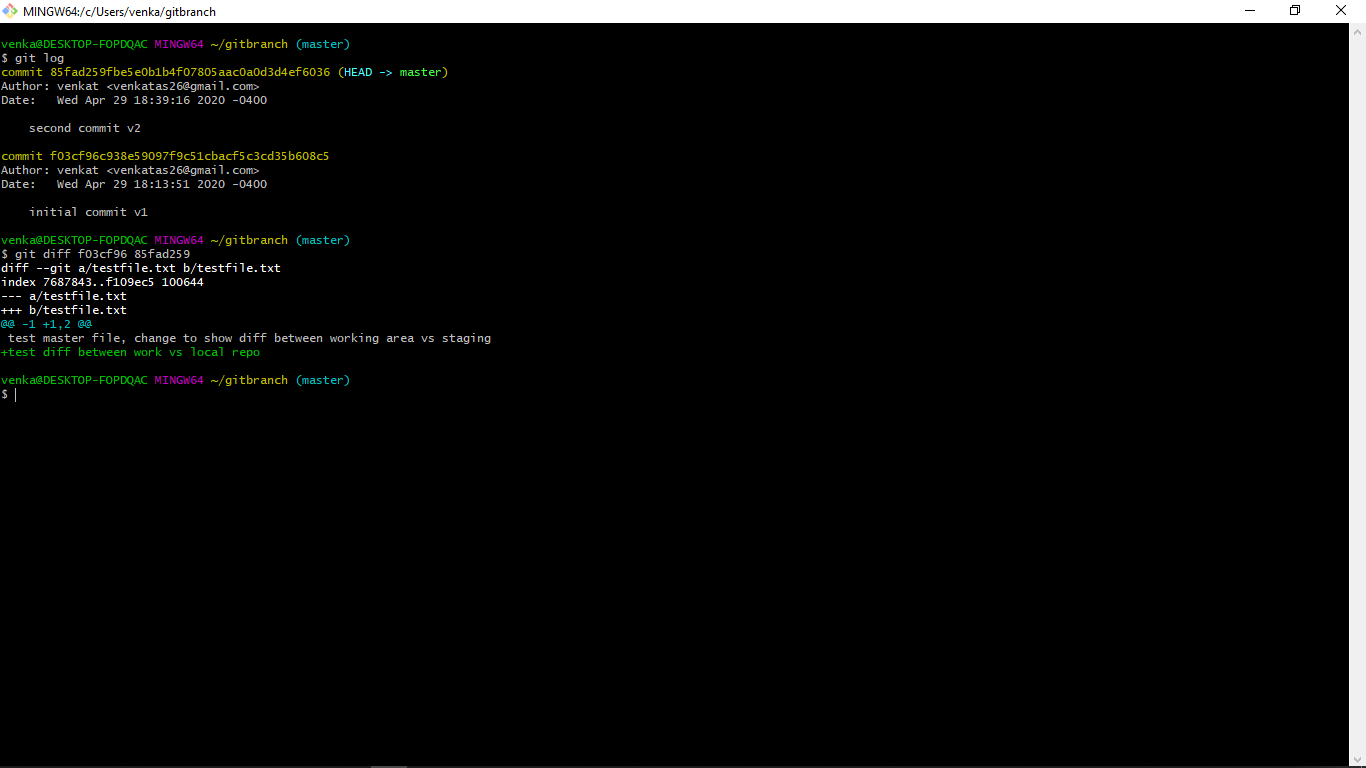


Committed the second change:

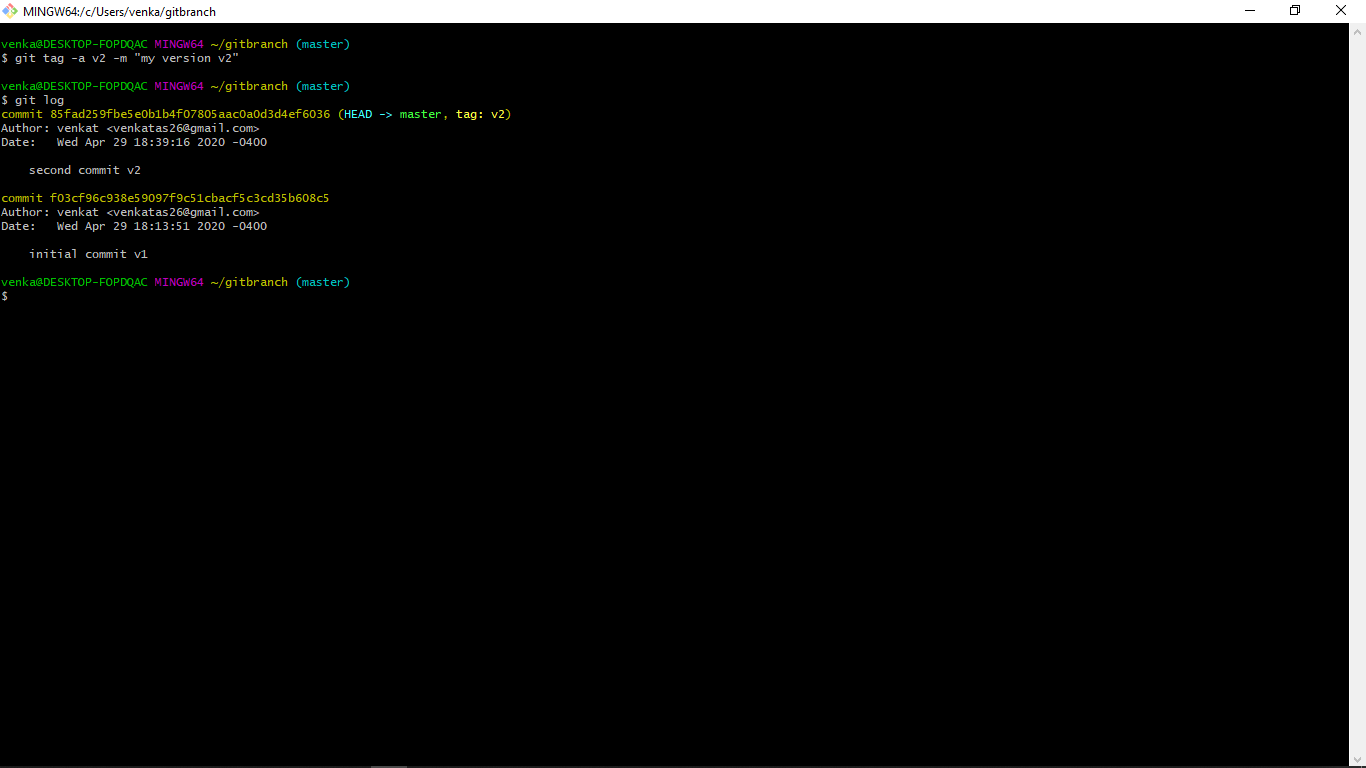




D. Difference between two commits



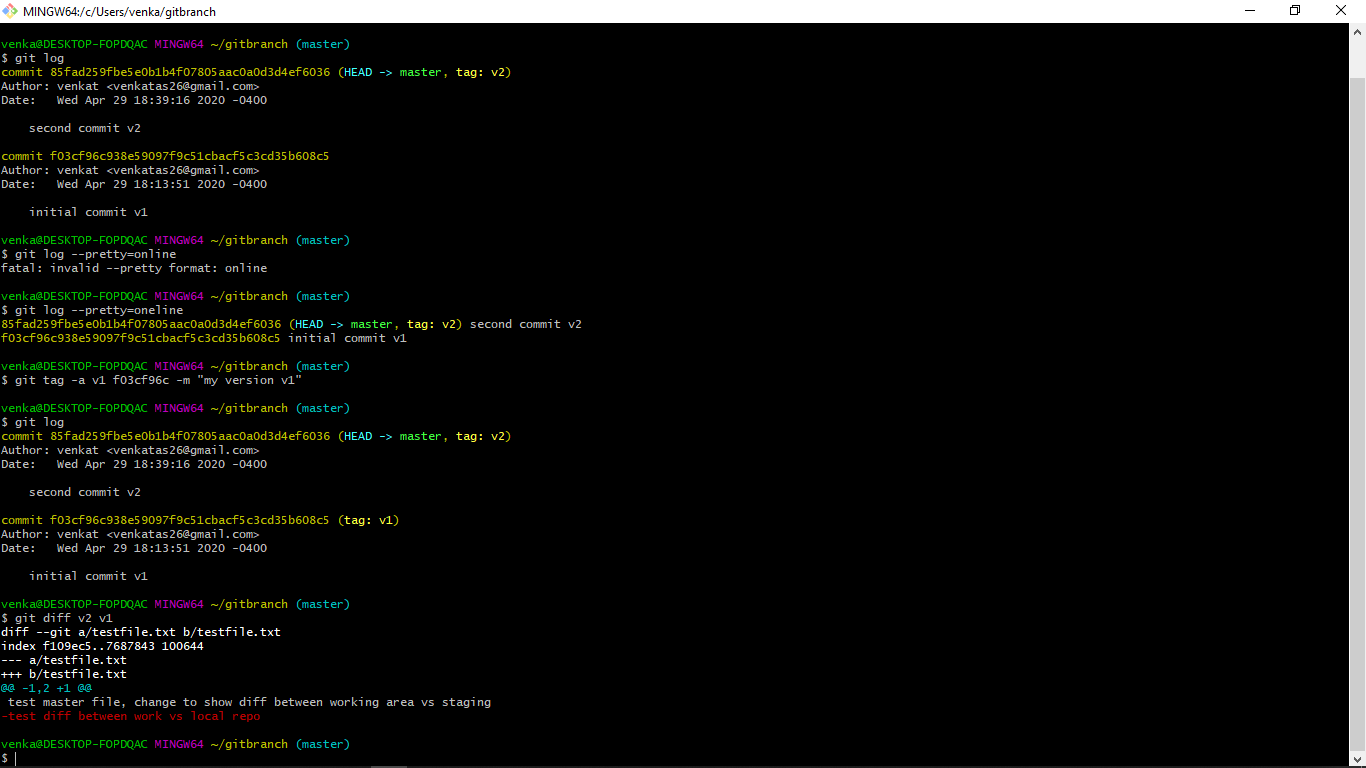
Added the tag to second commit



Added tag to first commit:

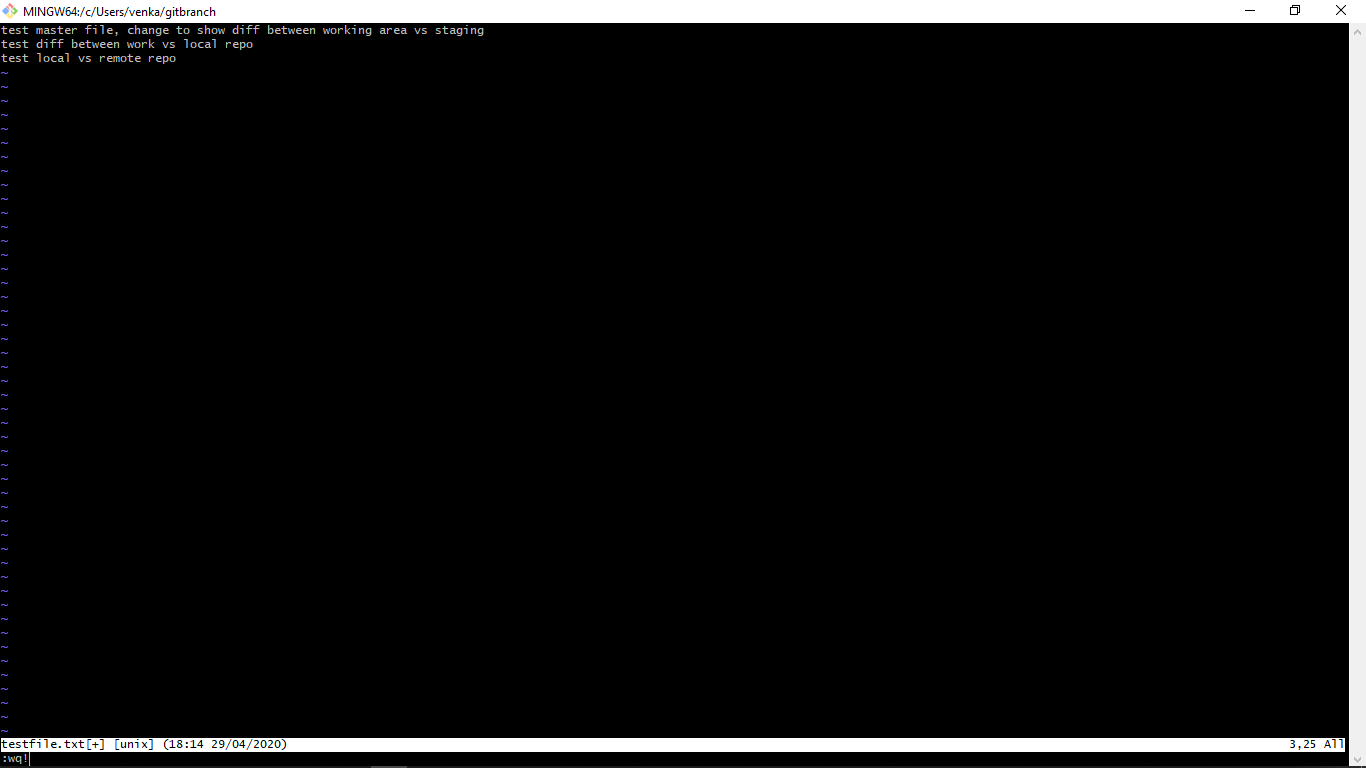


E. Diff between two tags:

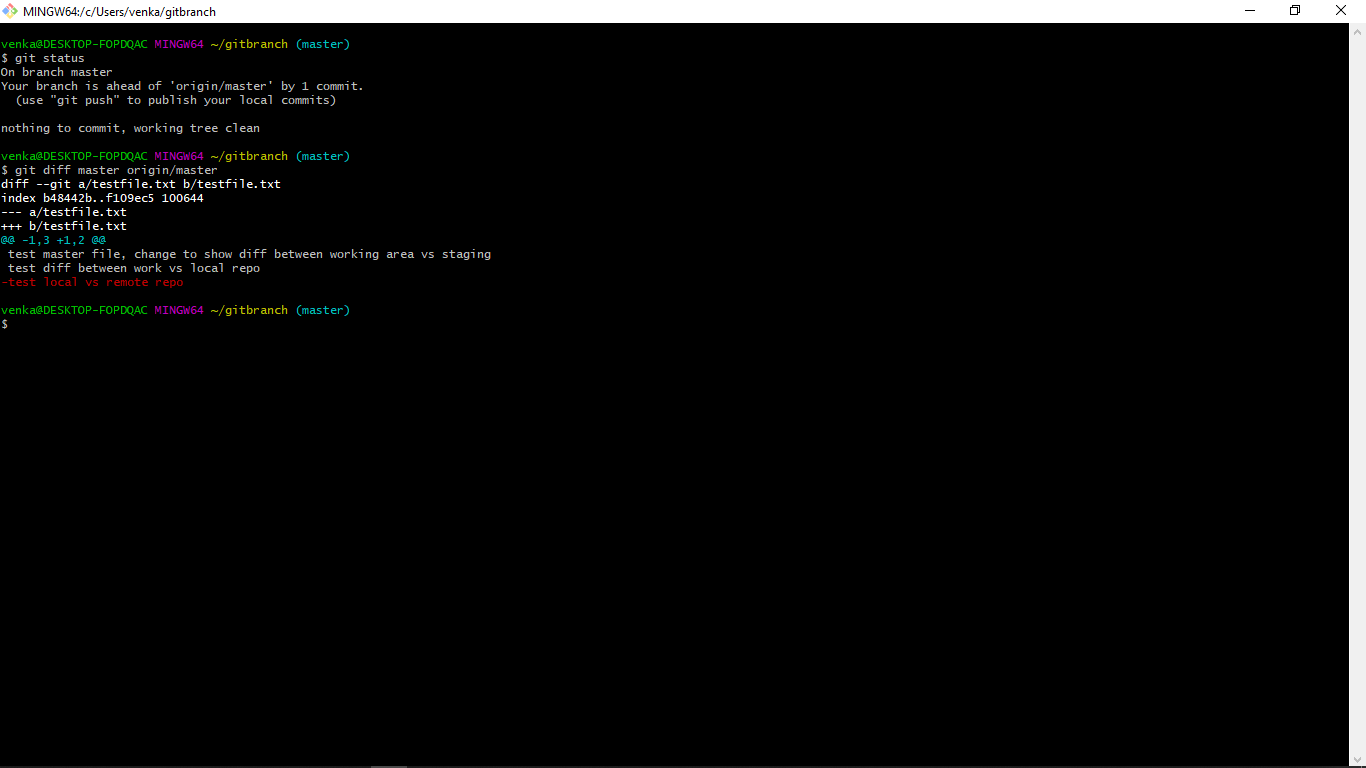


F. Local vs remote repository

Changed local repo file and did a local commit:



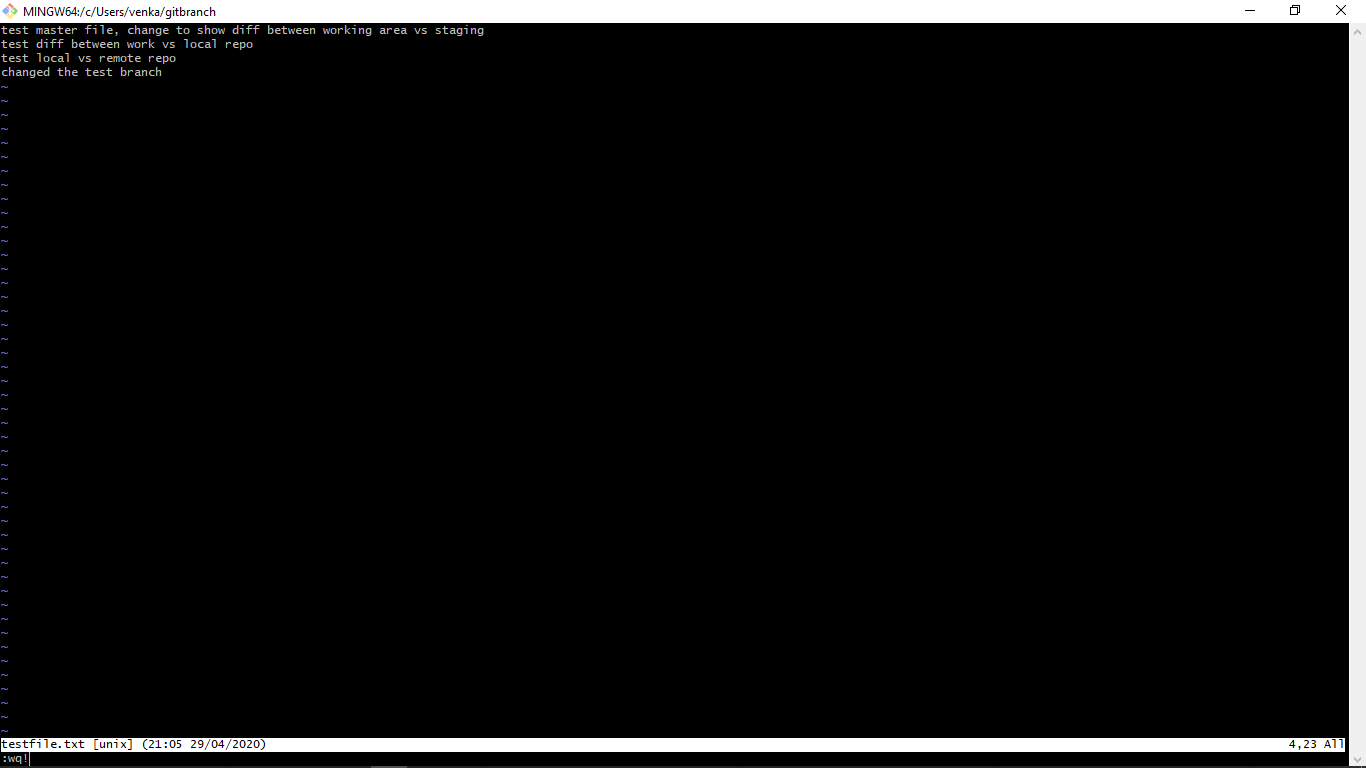
Did “git diff master origin/master”

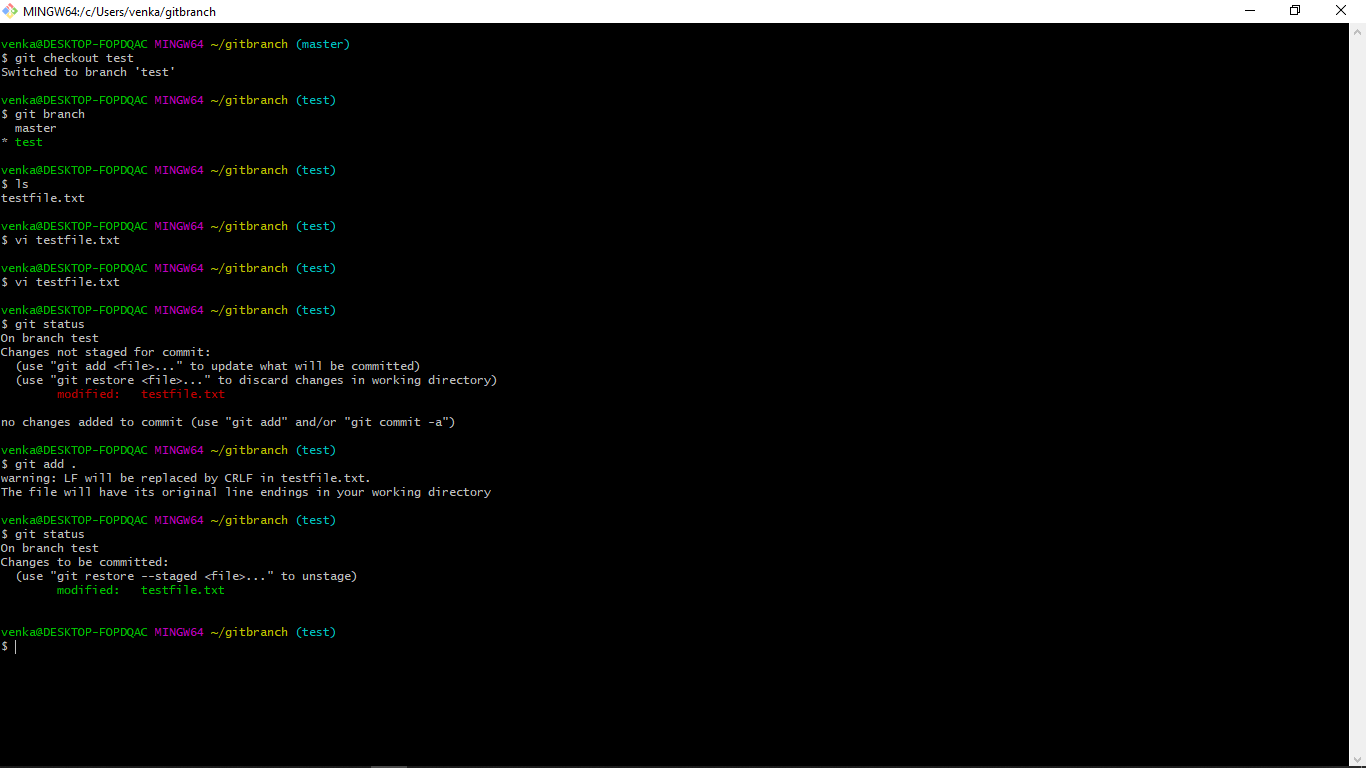


Moved the head to “Test” branch

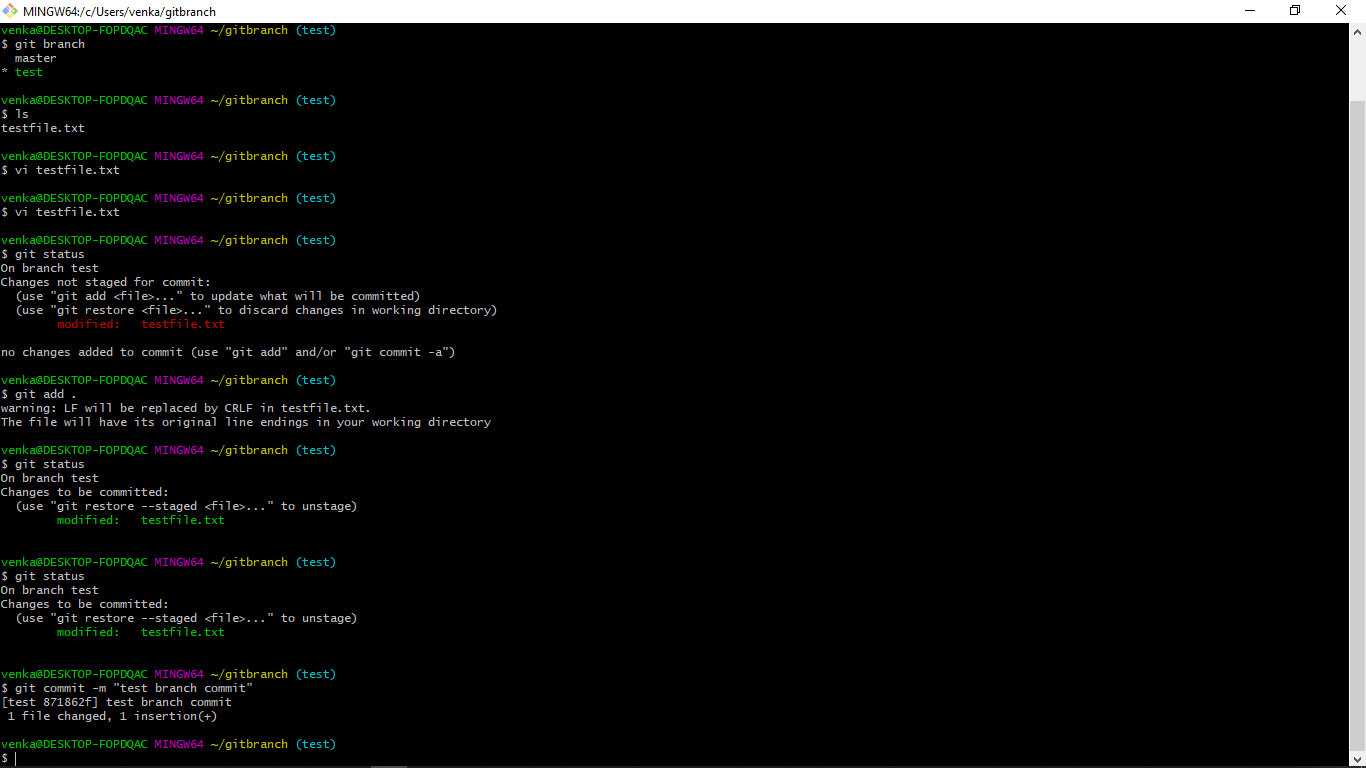


Changed the file in test branch:

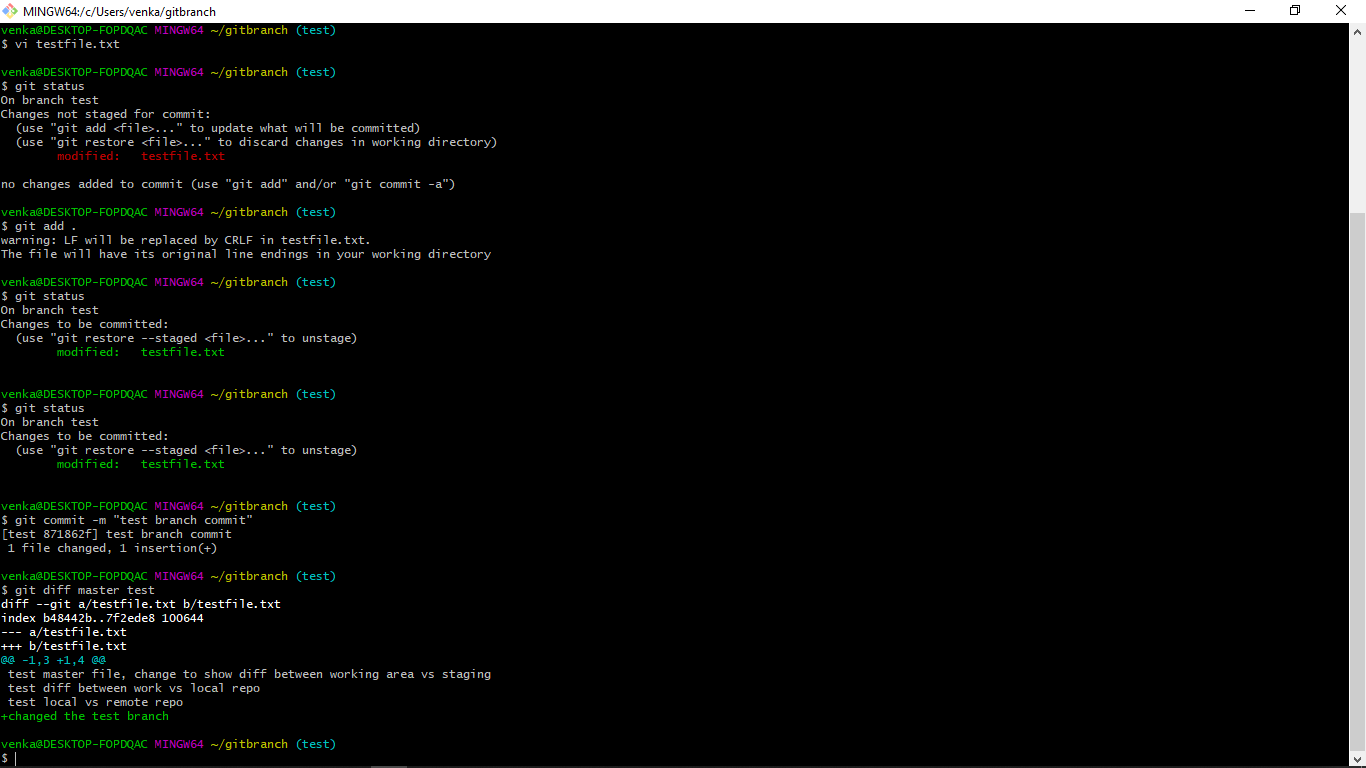




Committed the changes to test branch:

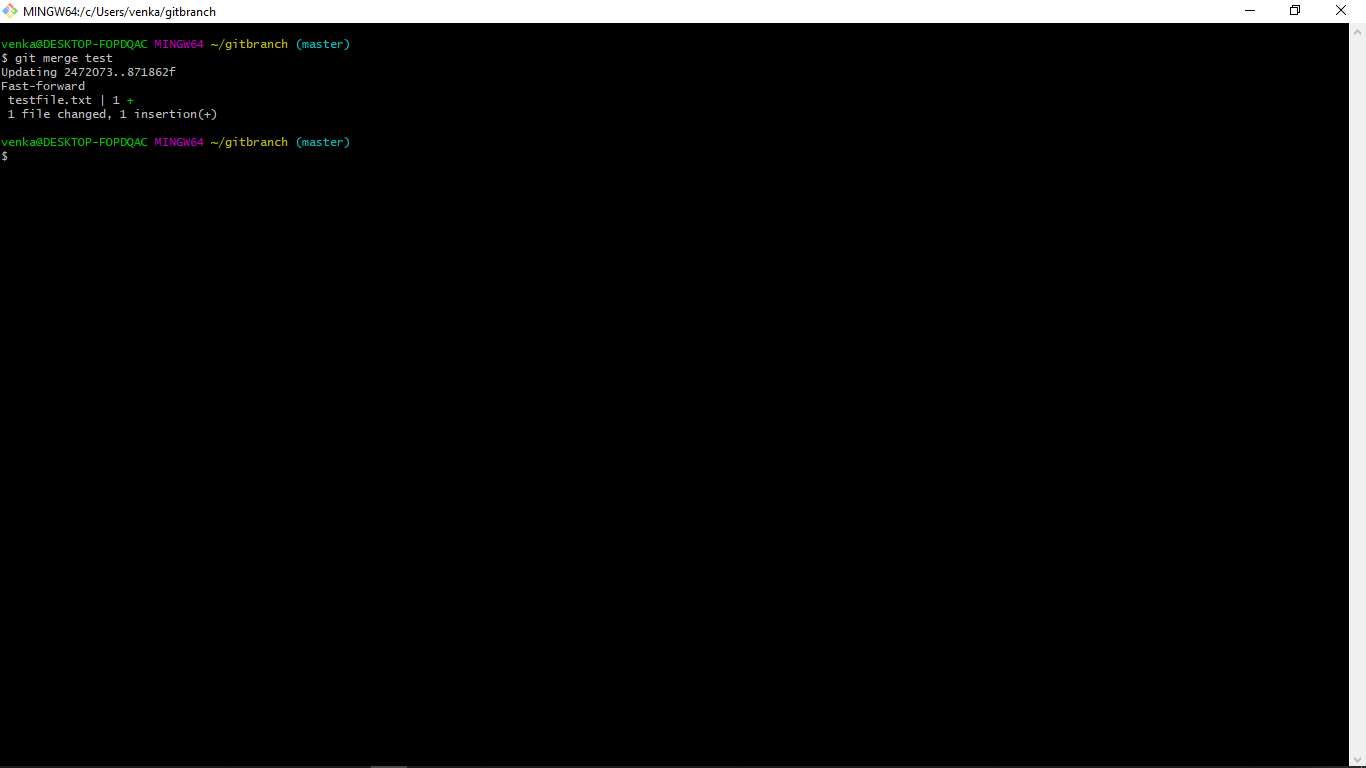


“Git Diff master test” shows the difference between master branch and test branch



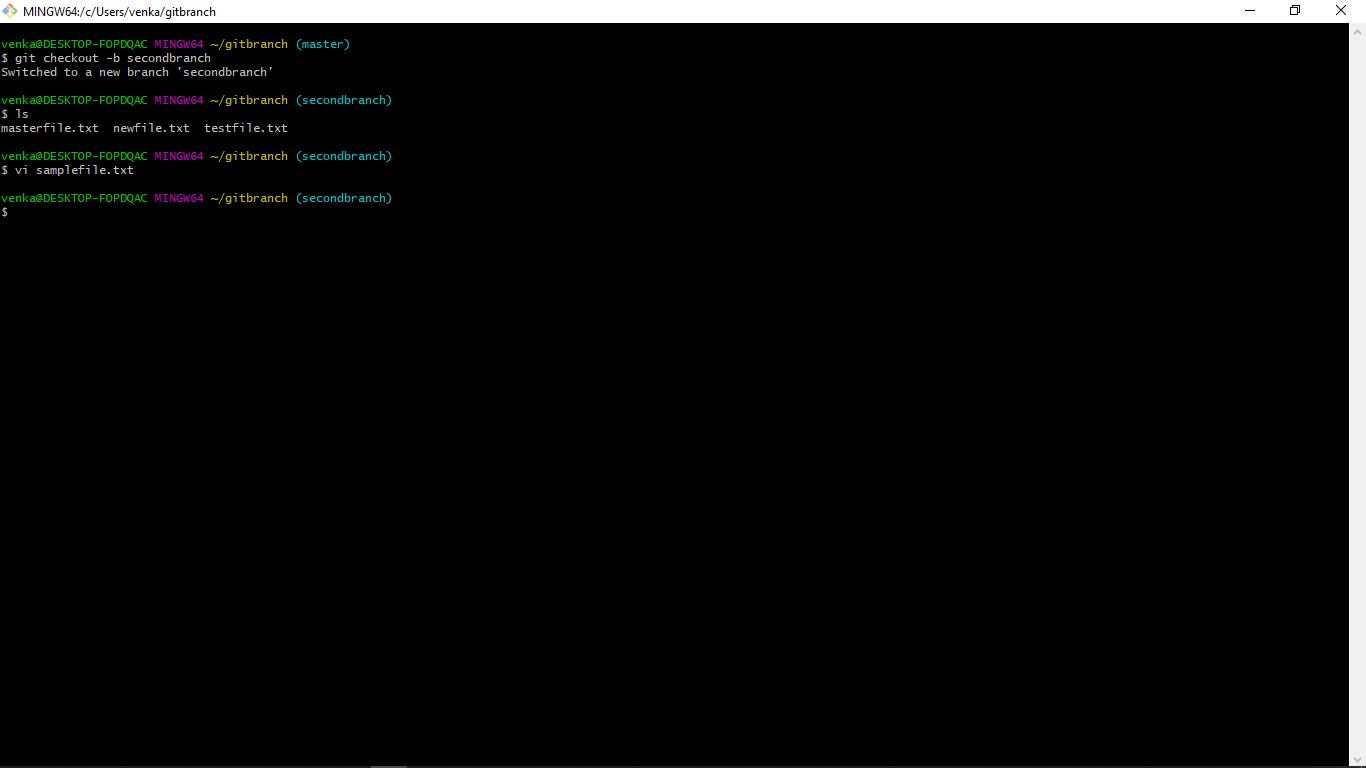
10. Merge the changes from test branch to master branch

1. Fast forward merge

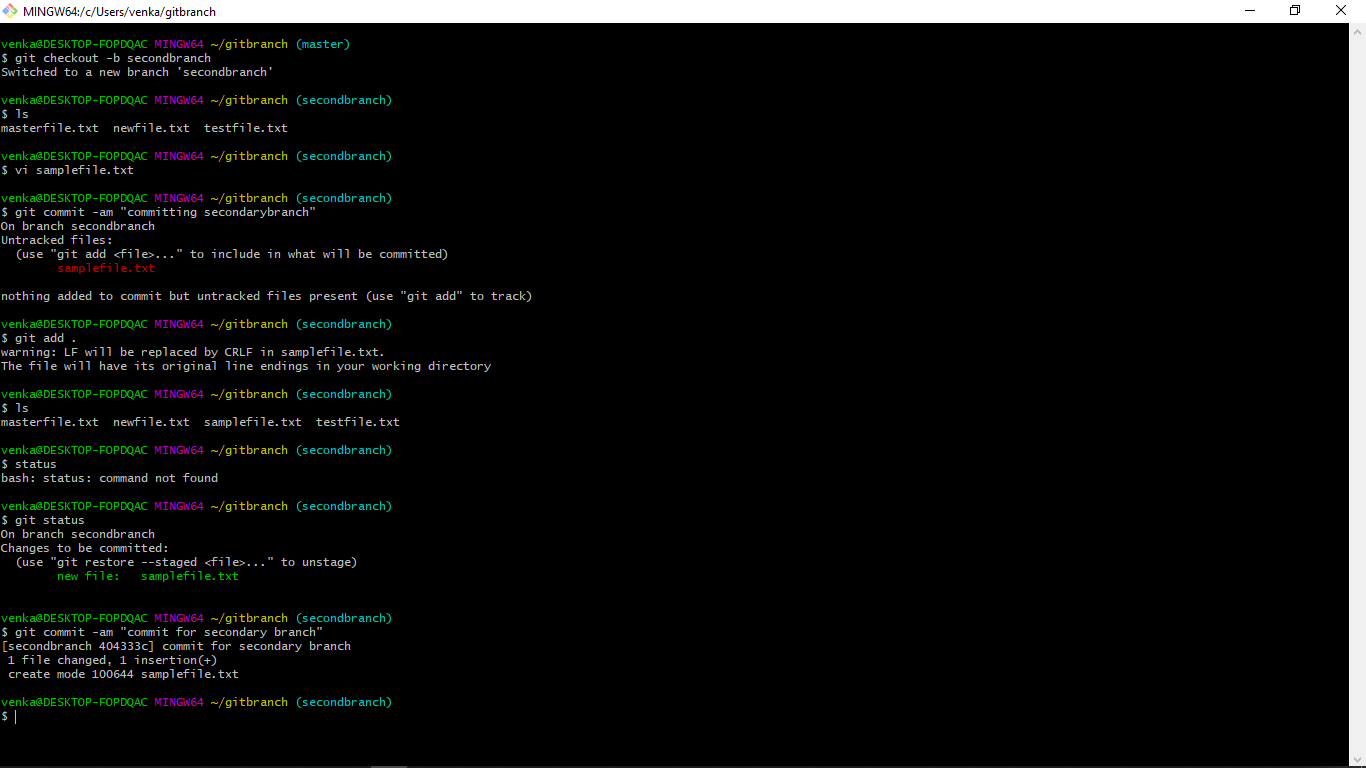


1. Disabling fastforward merge

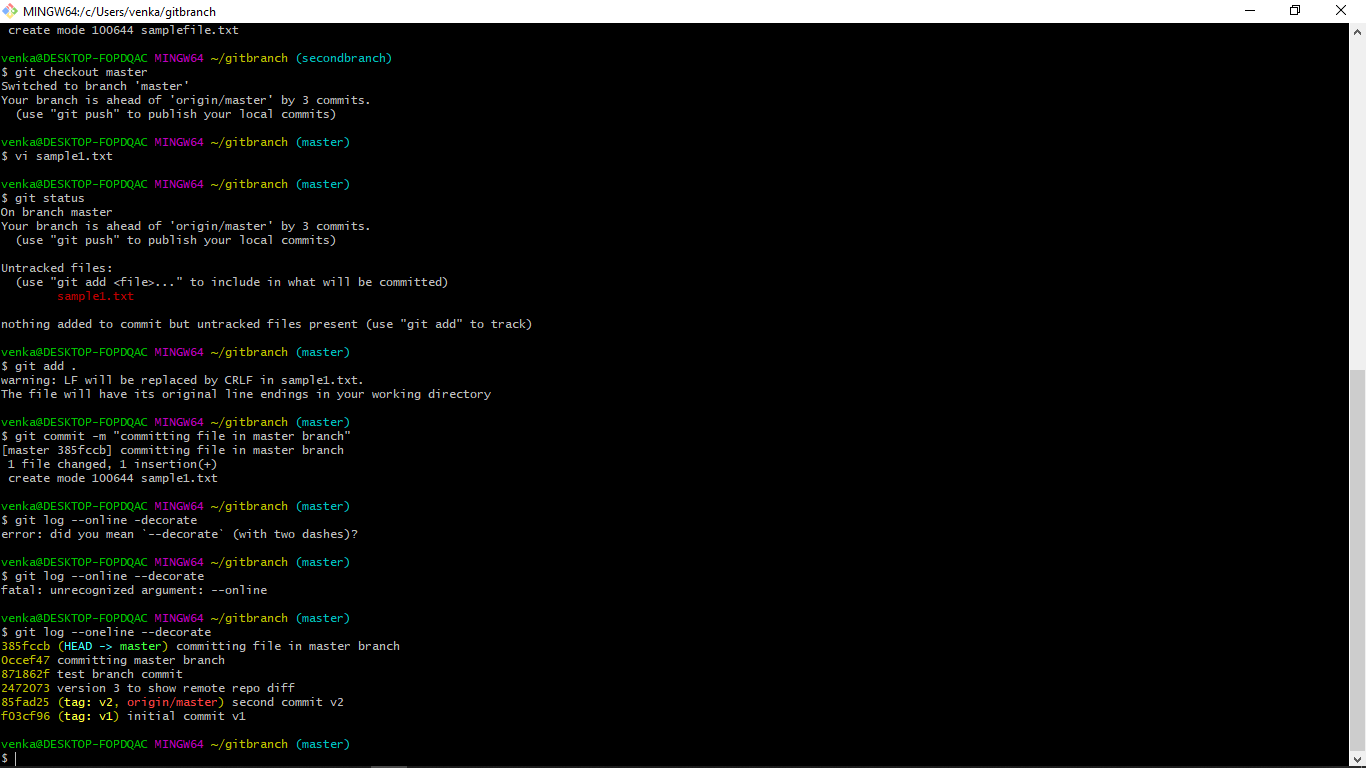
Created a new branch and added a new textfile

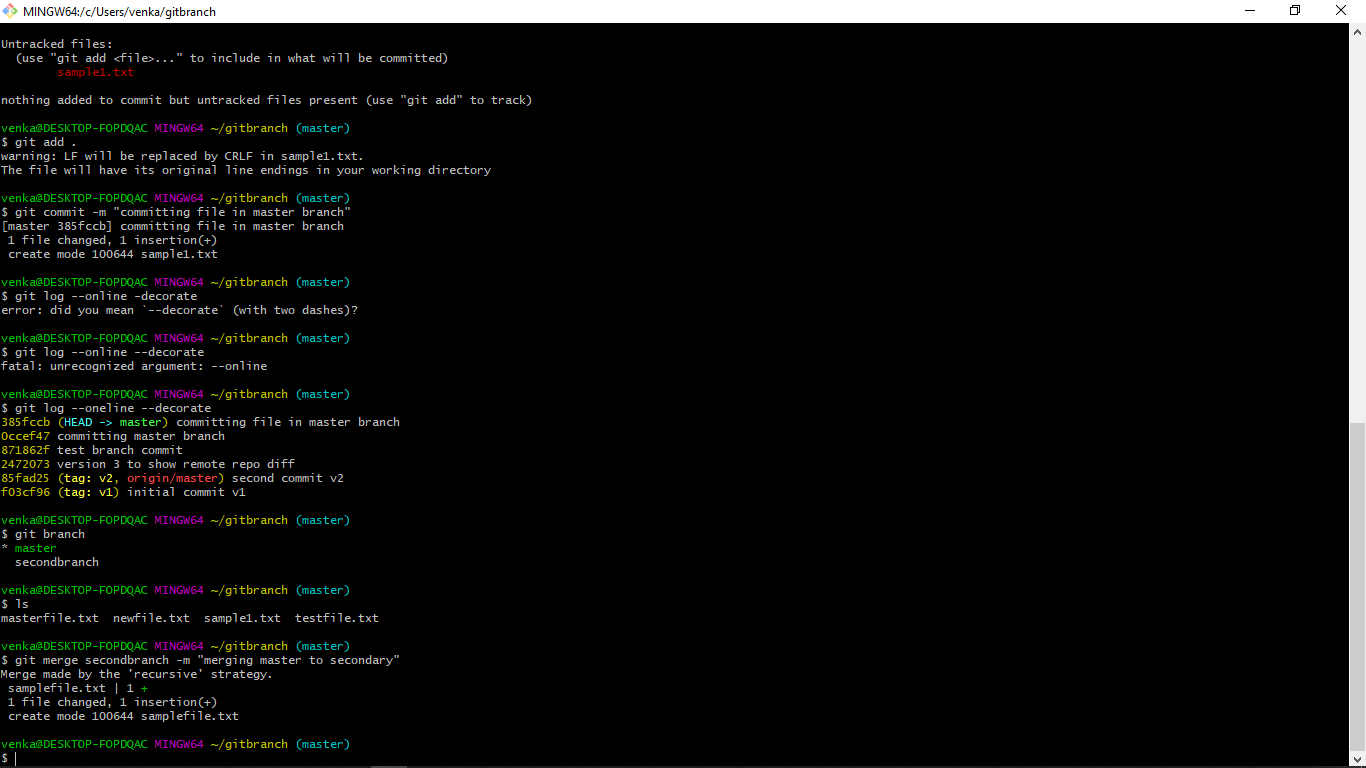






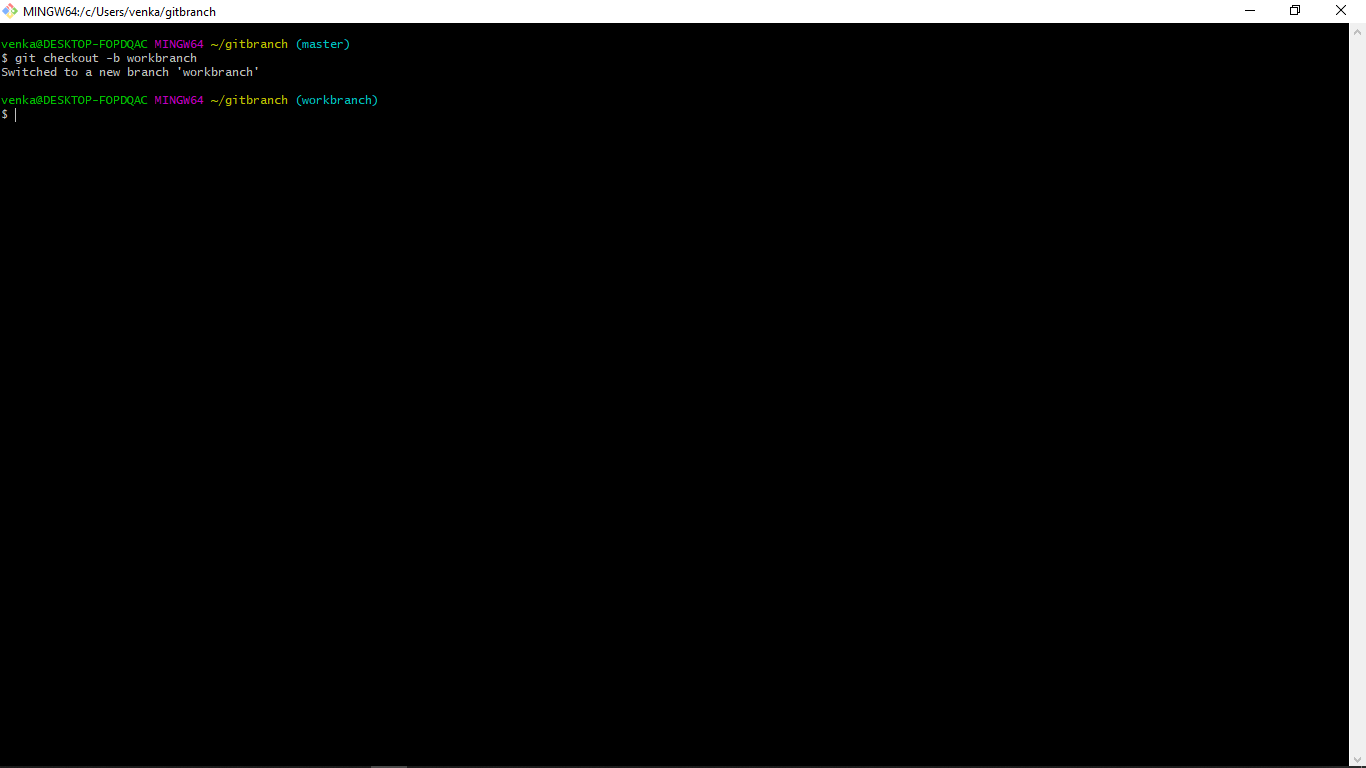




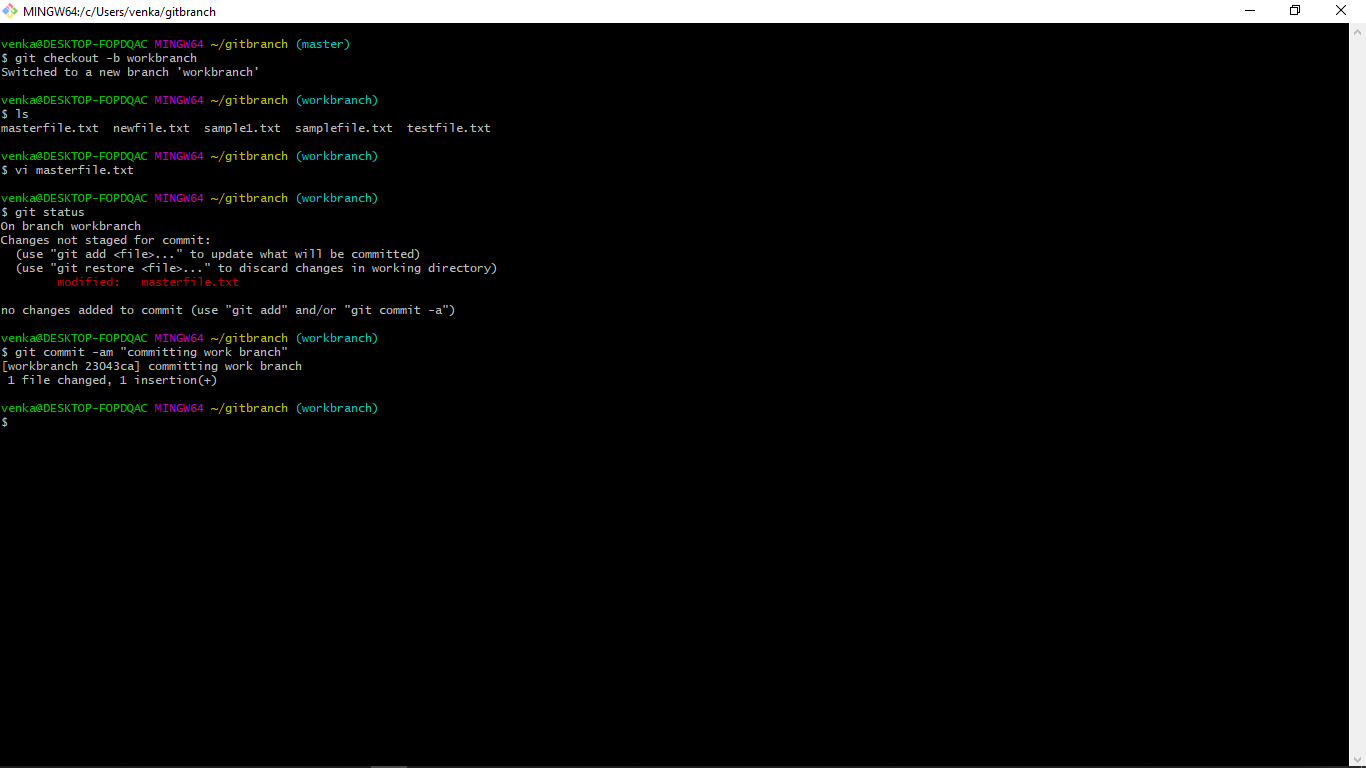


c) In fast forward, the history of branch merges are ignored and the final merge will have just single commit and this is useful while working on production like environment. No fastforward is useful for development like environment where all the changes from sub branches from different developers are merged into the master branch with all merge history for logging purpose.

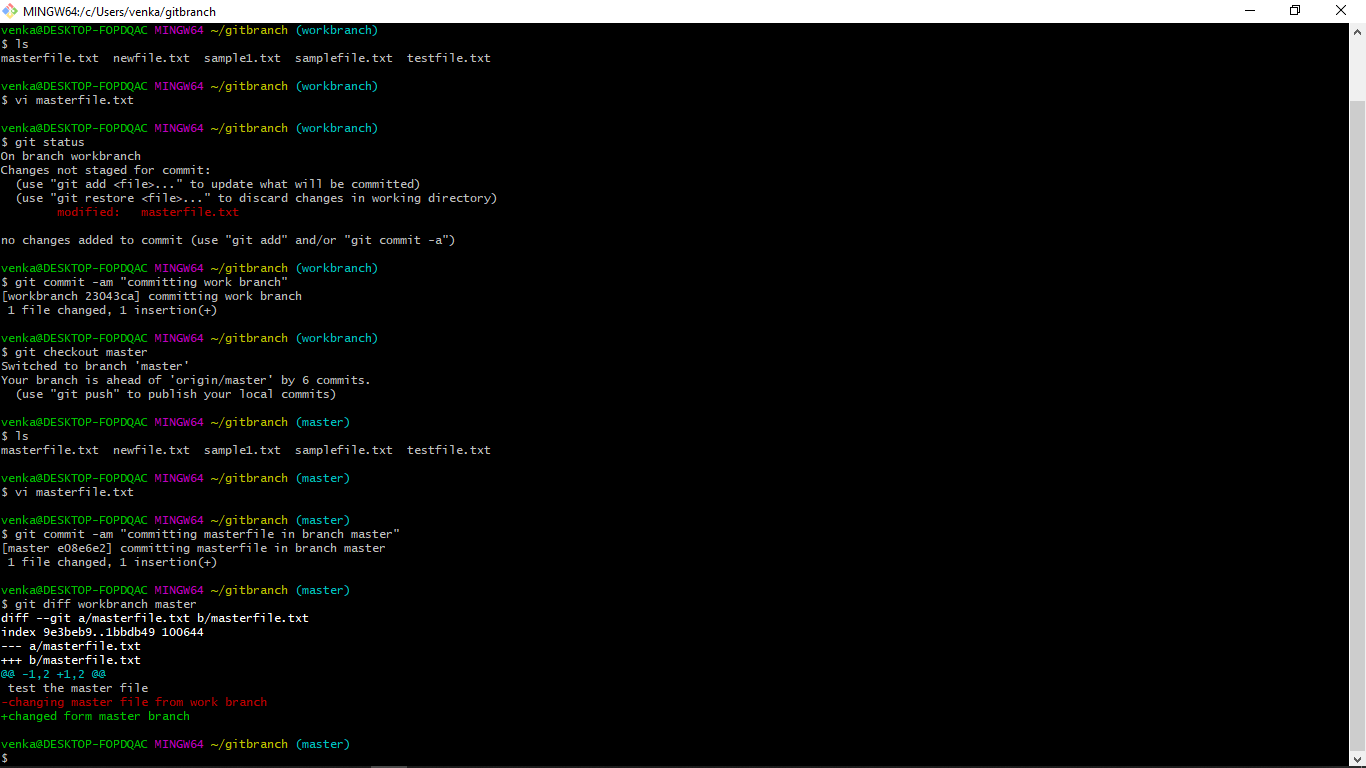
11. Merge conflict situation



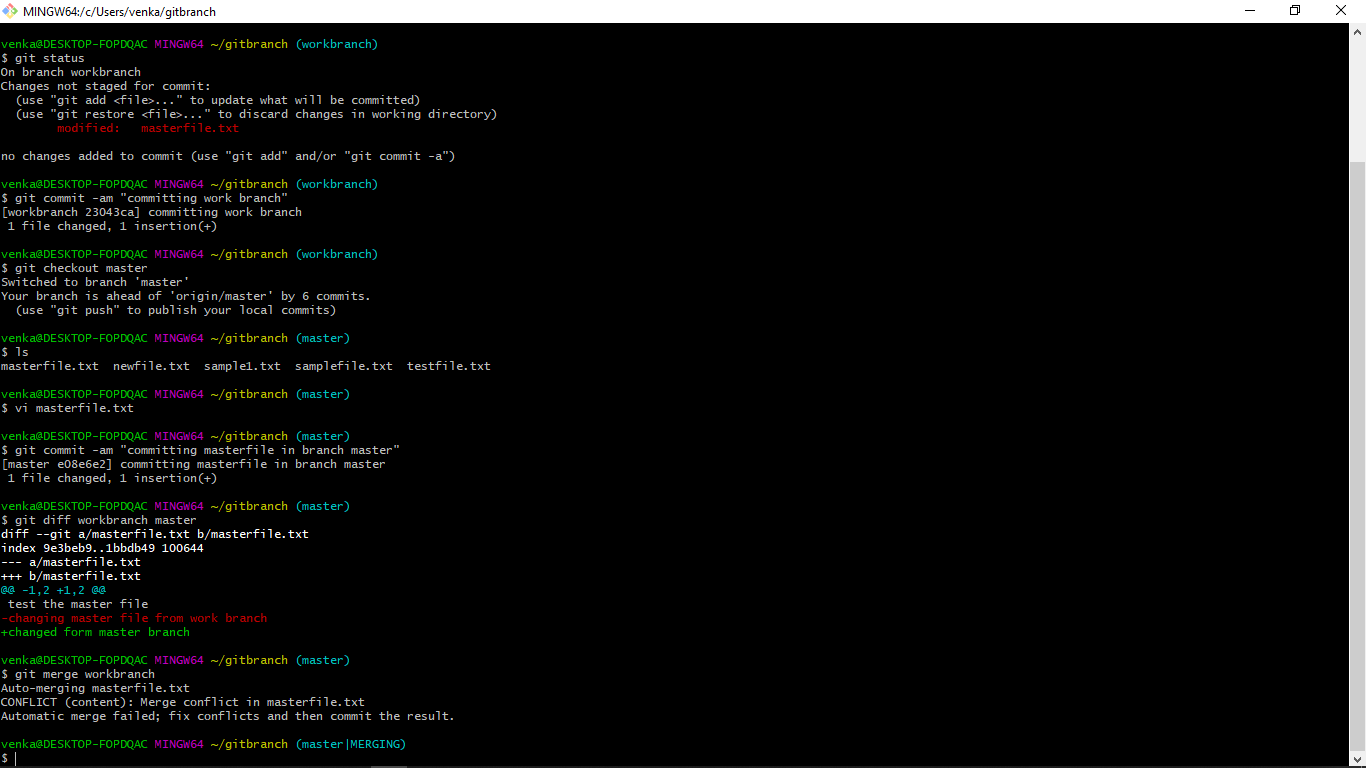


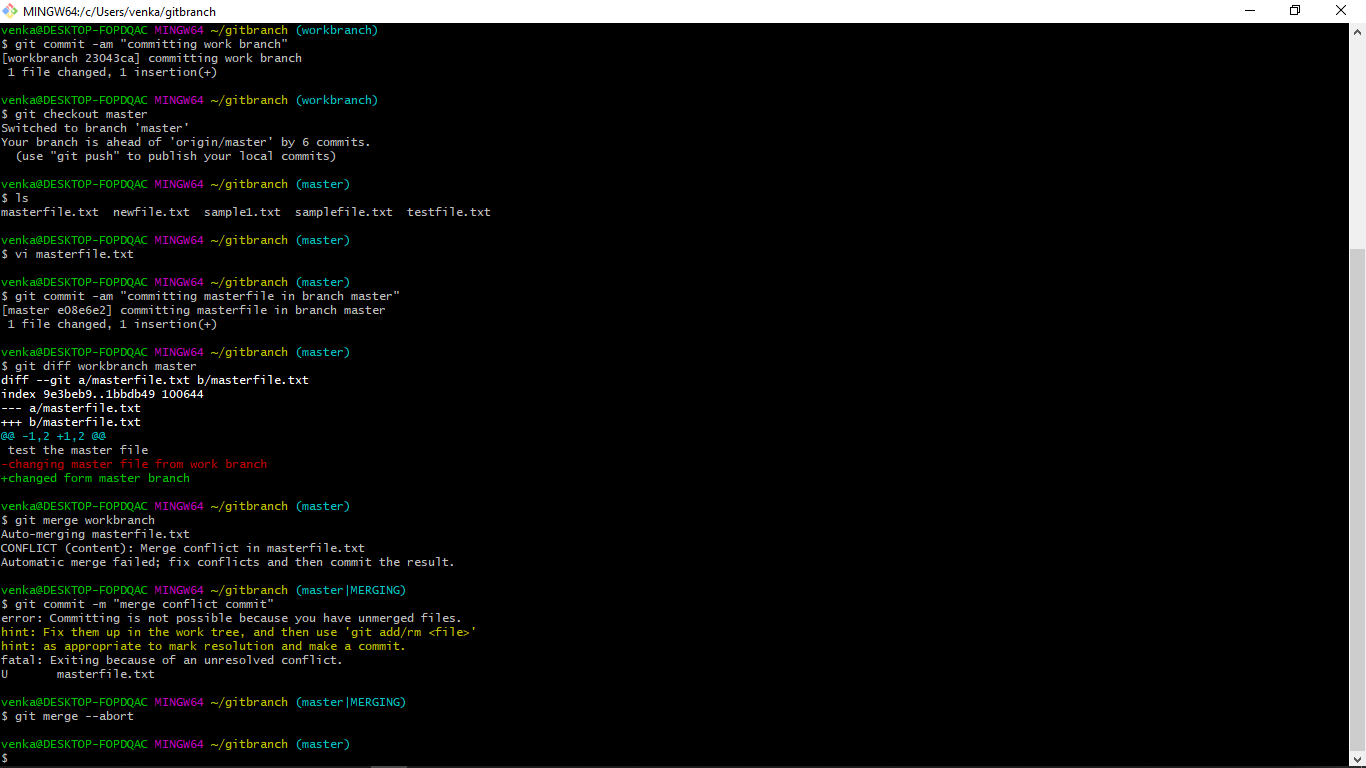






Merge conflict:

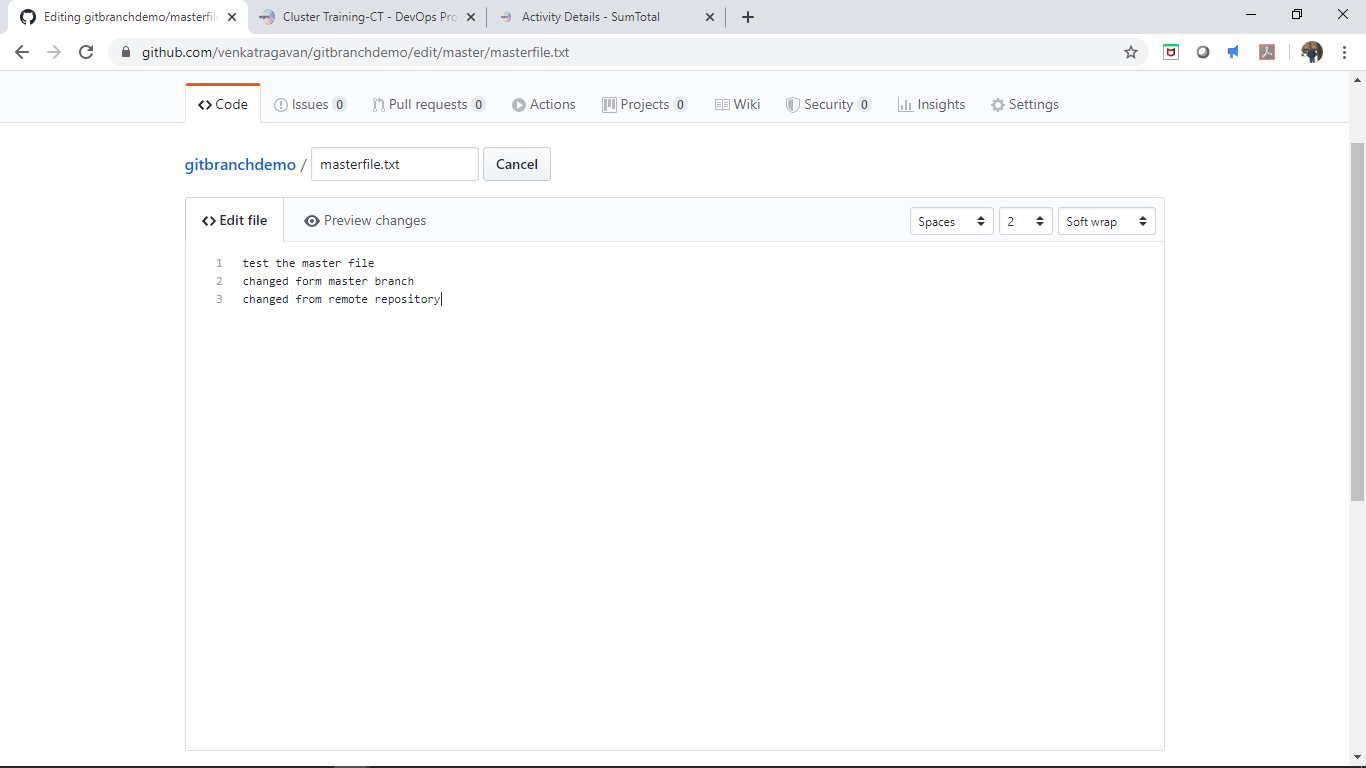


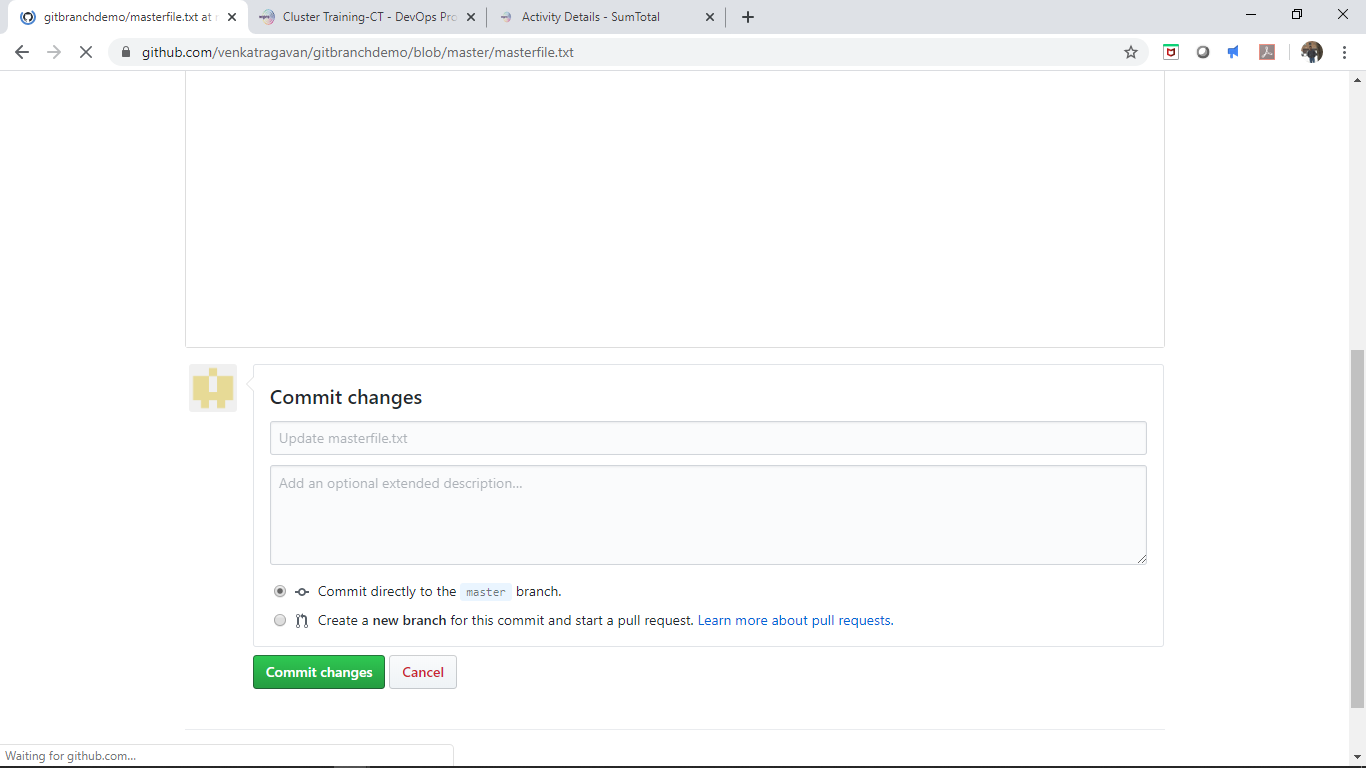


13. Fetch, clone and pull

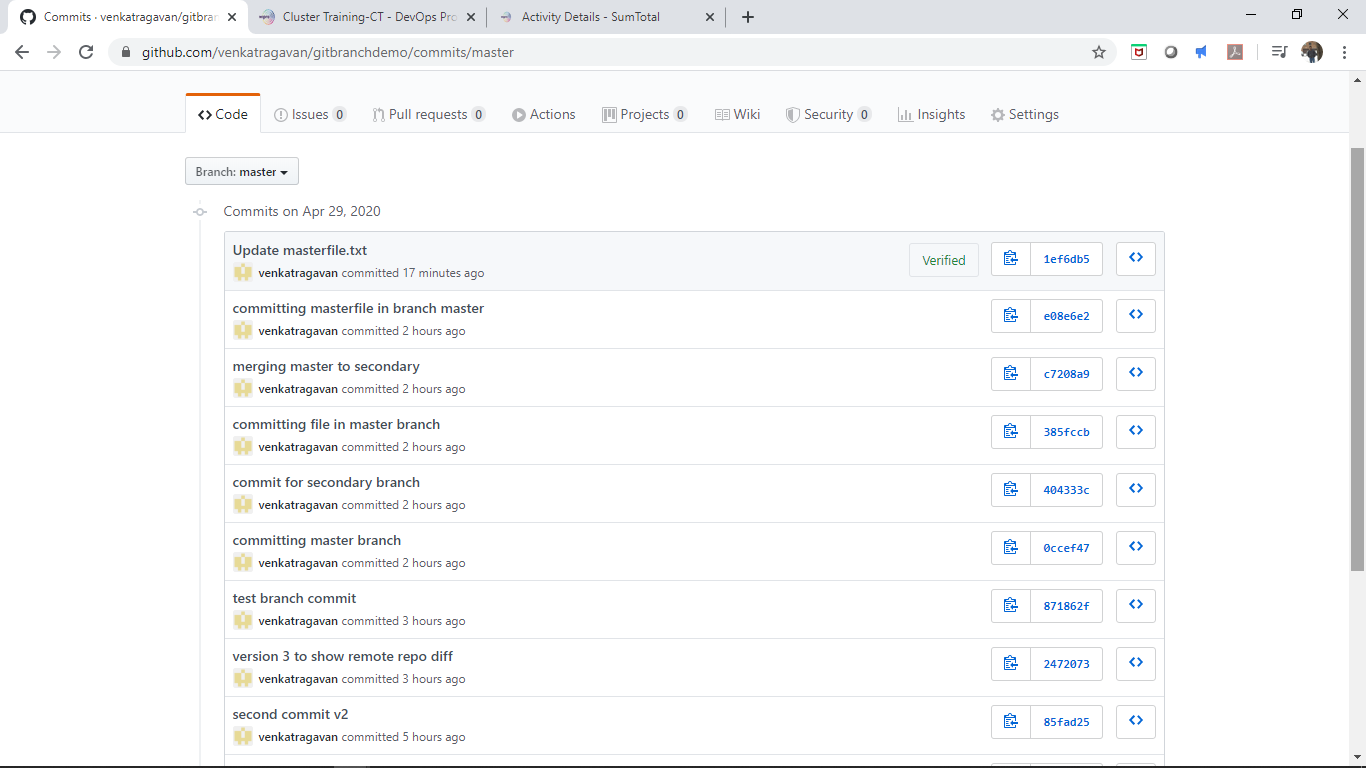
Fetch:

Did some changes to remote github and committed in github:

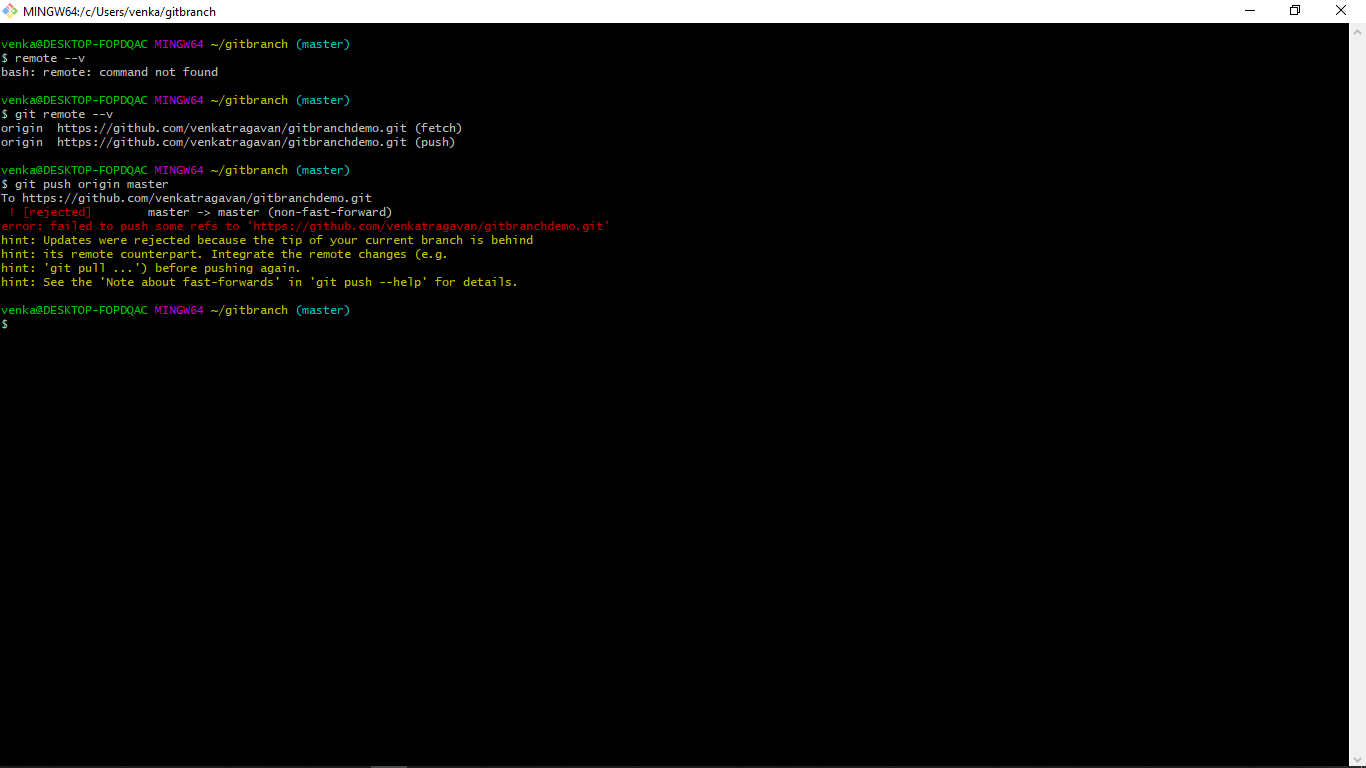




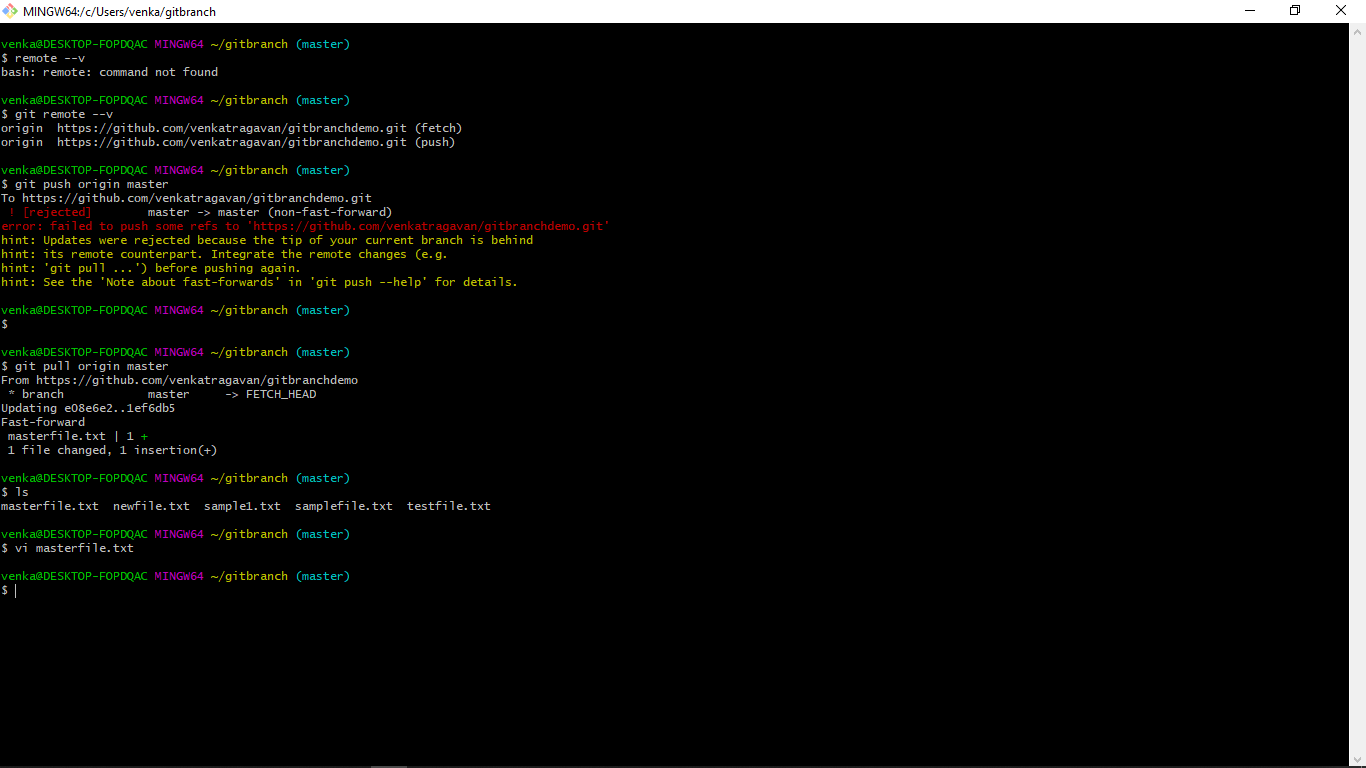
New commit is in remote repo on master branch:



Will do a git push on local git, which gives error as the remote repo is one step ahead of the local branch.



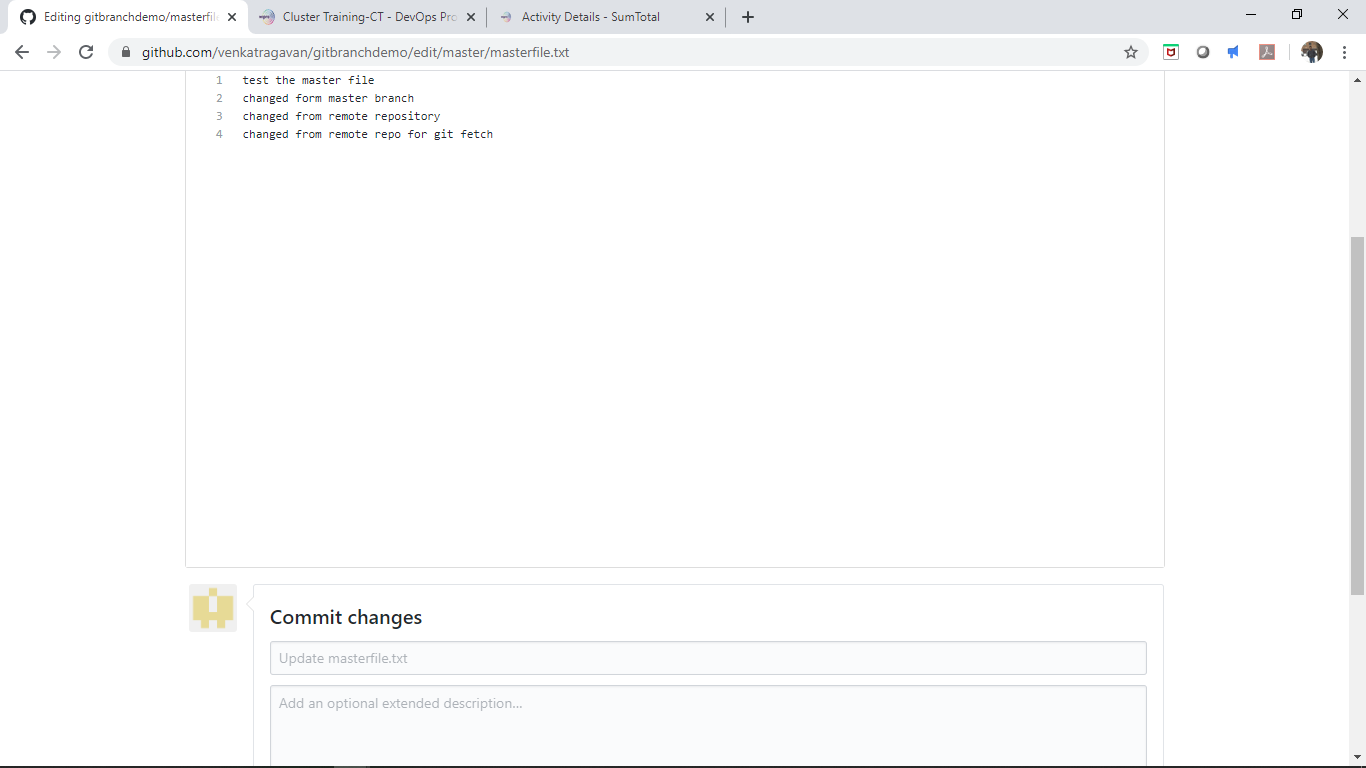
Now git pull will bring the changes from remote repository and sync(merge) in with the local repo and replace the changes in local repo with remote repository.

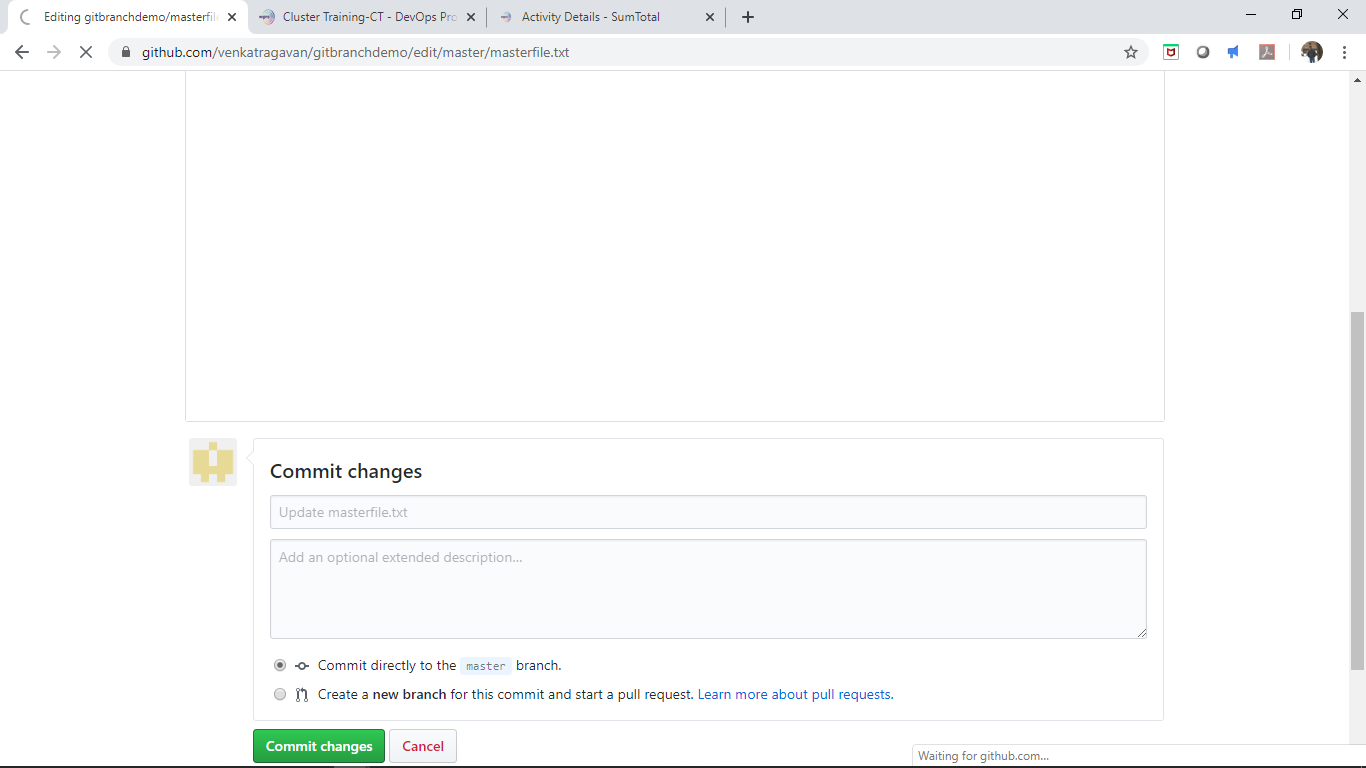




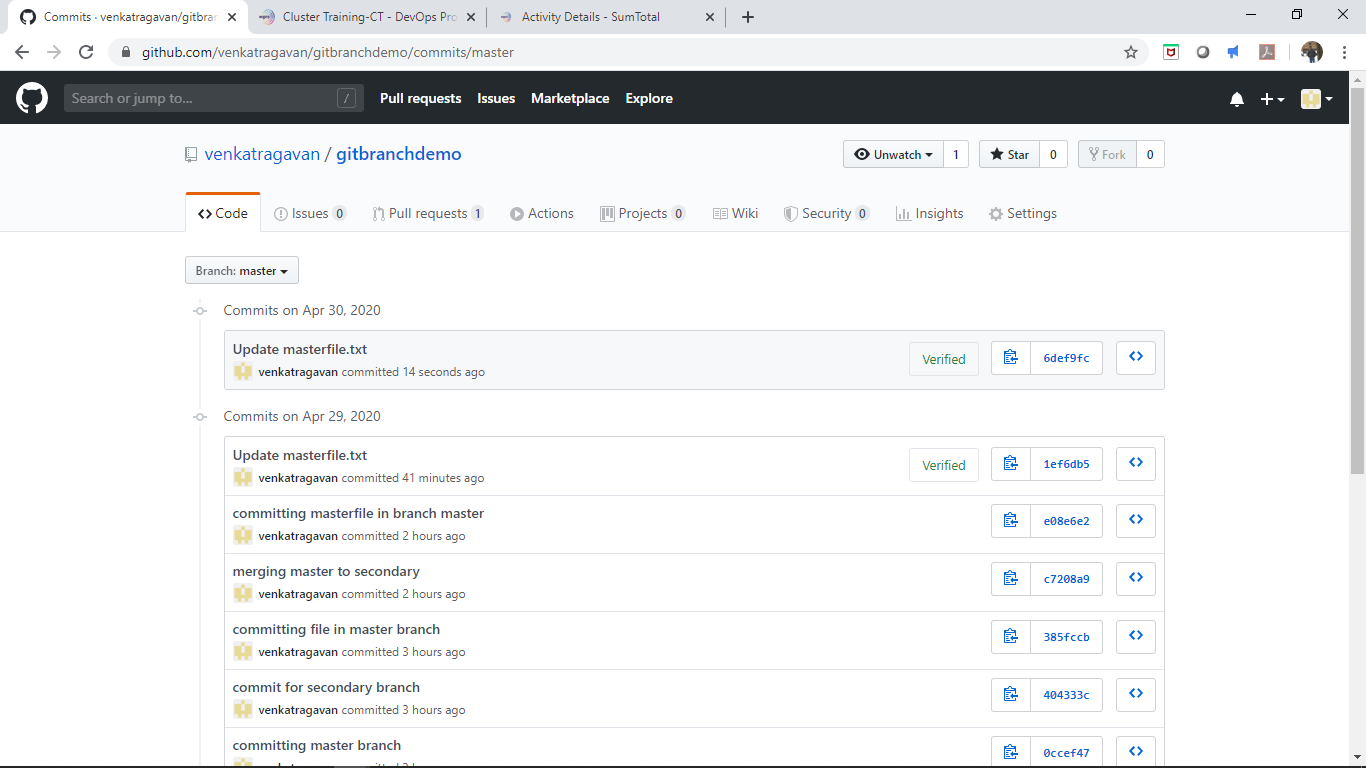
Git Fetch:

Created a new branch in remote repo as shown below with a new file:





We can see remote repo is committed:



The git fetch, took a snapshot of the commits and new branch in remote repo and downloaded into local repo but it won’t replace the working copy. Git fetch won’t do a merge and keep the working copy as it is



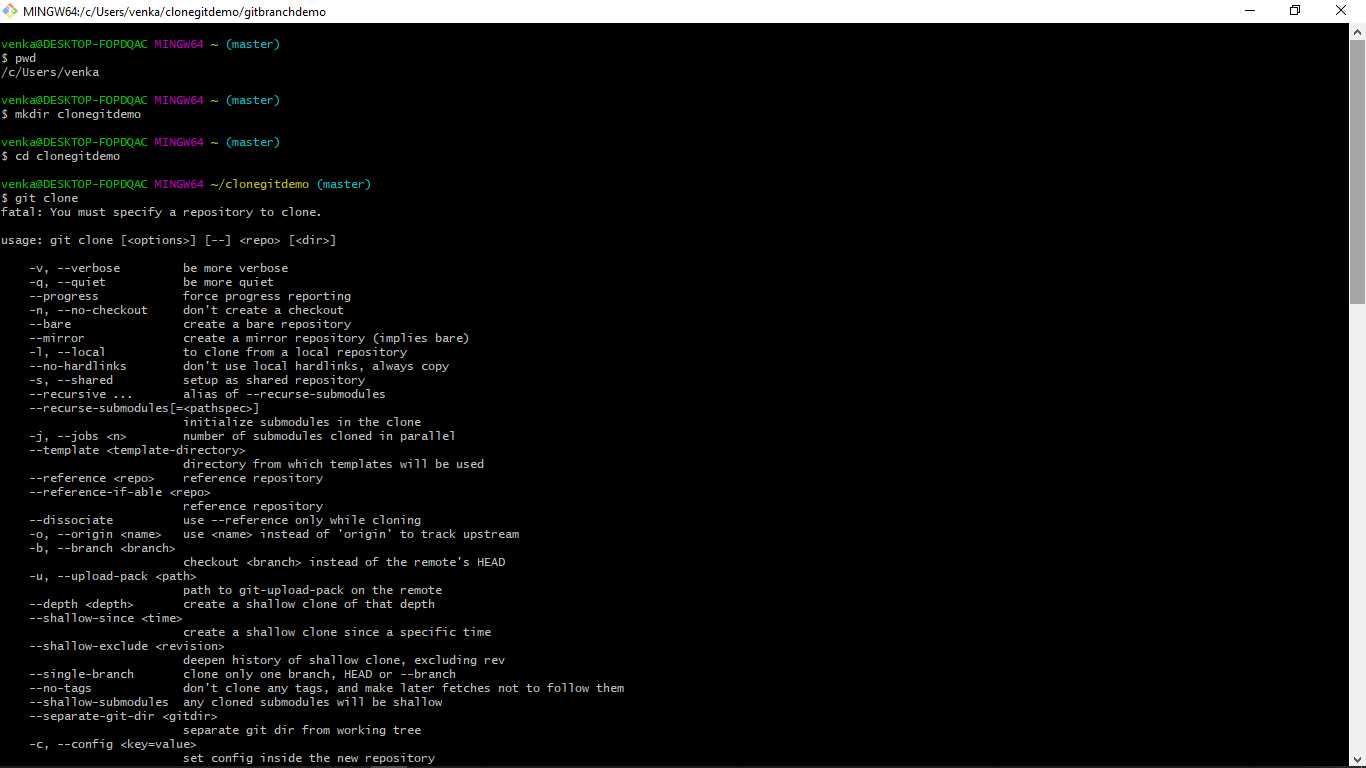
Did a checkout to remote/origin and we can see the changes done in remote repo in github is available in local repo at detached state:

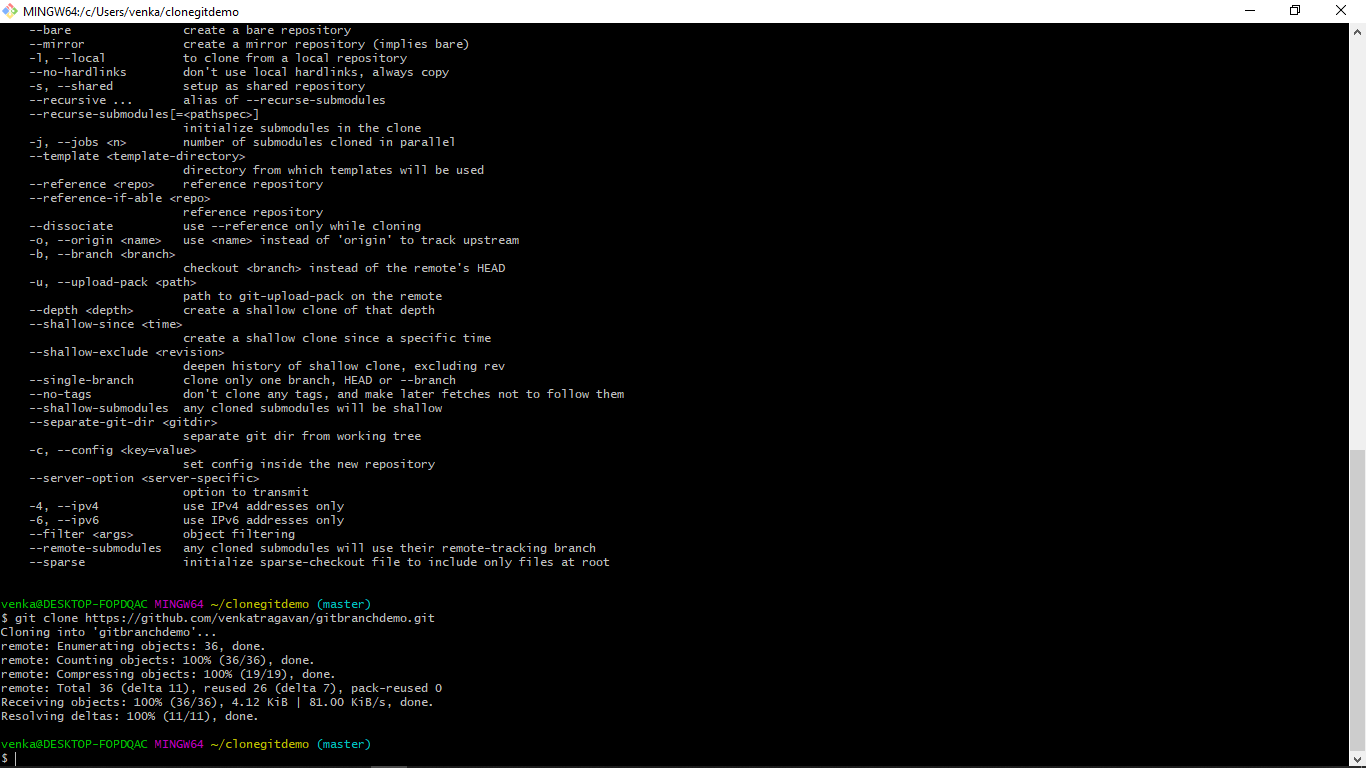




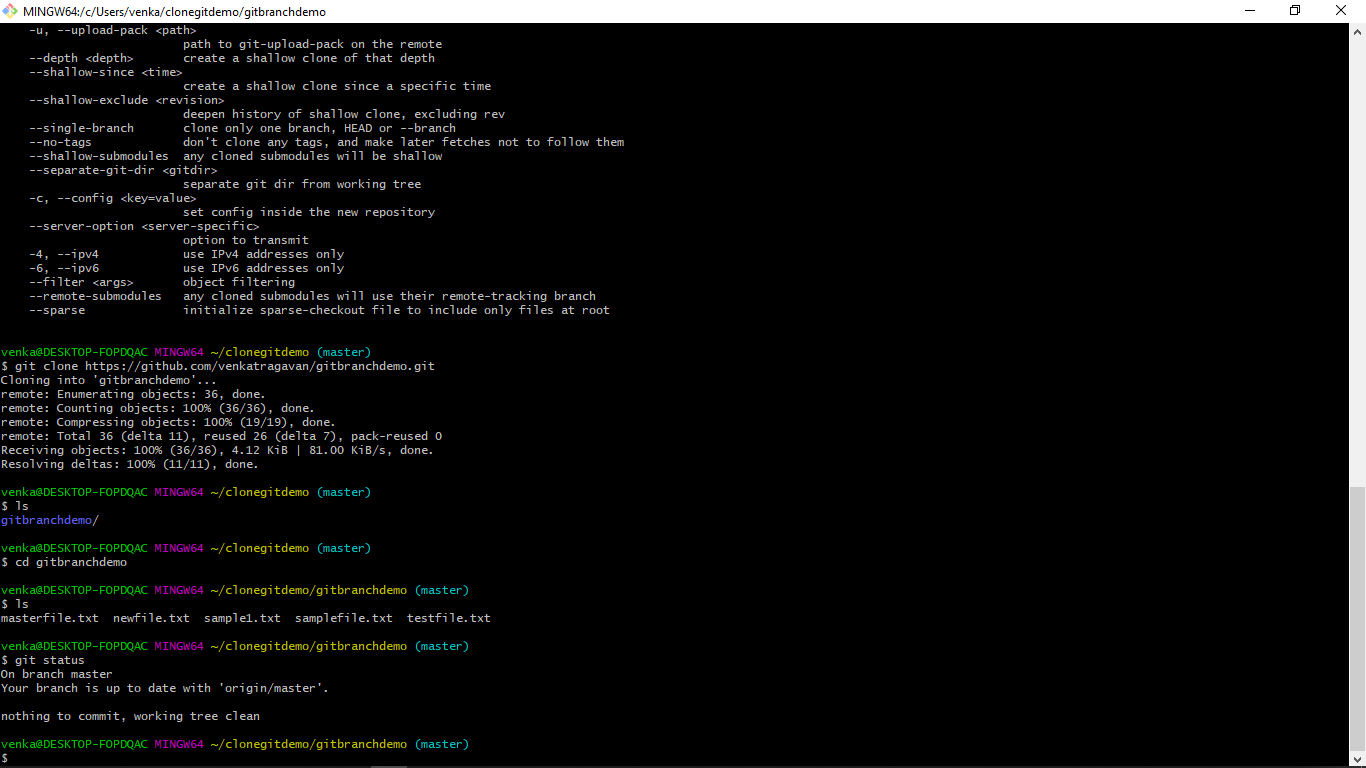
Git Clone:

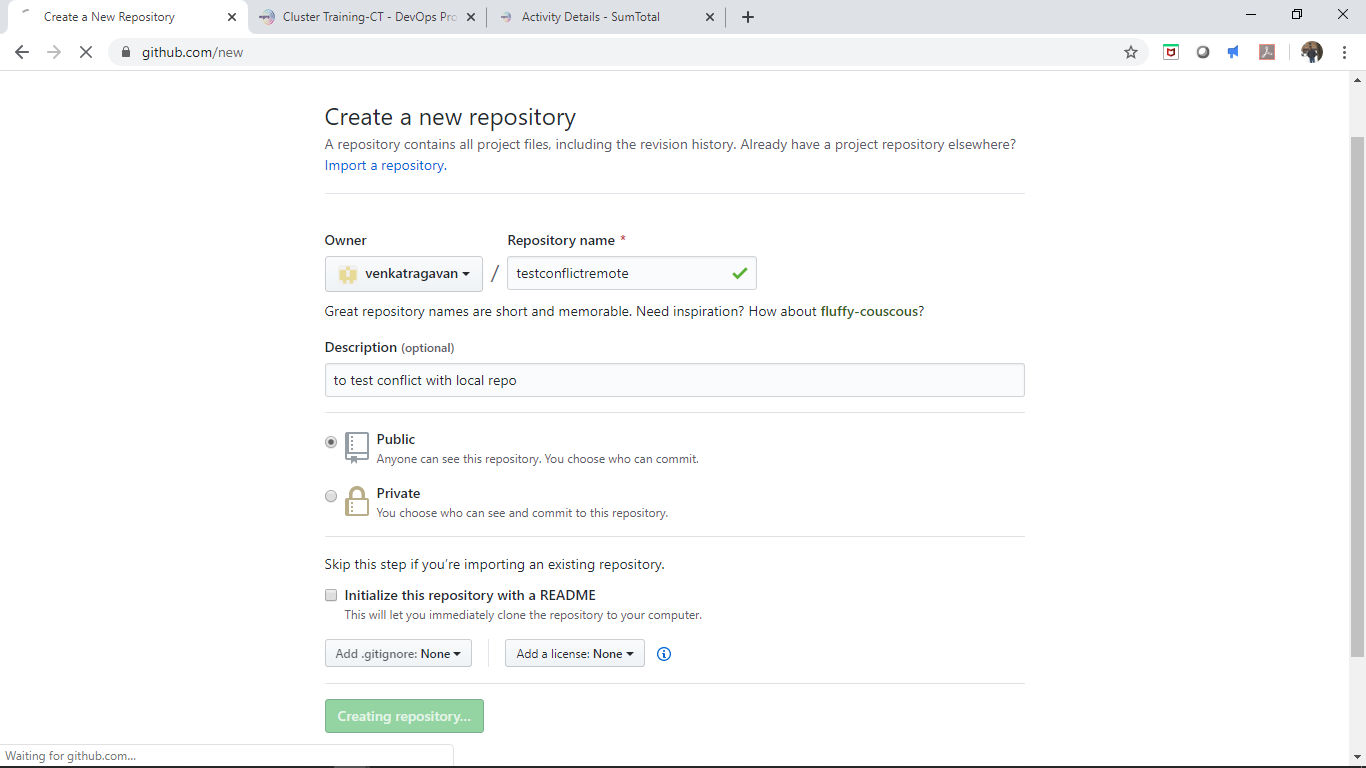
Git clone is used to download a project from remote repository into our local directory to create a fresh copy.

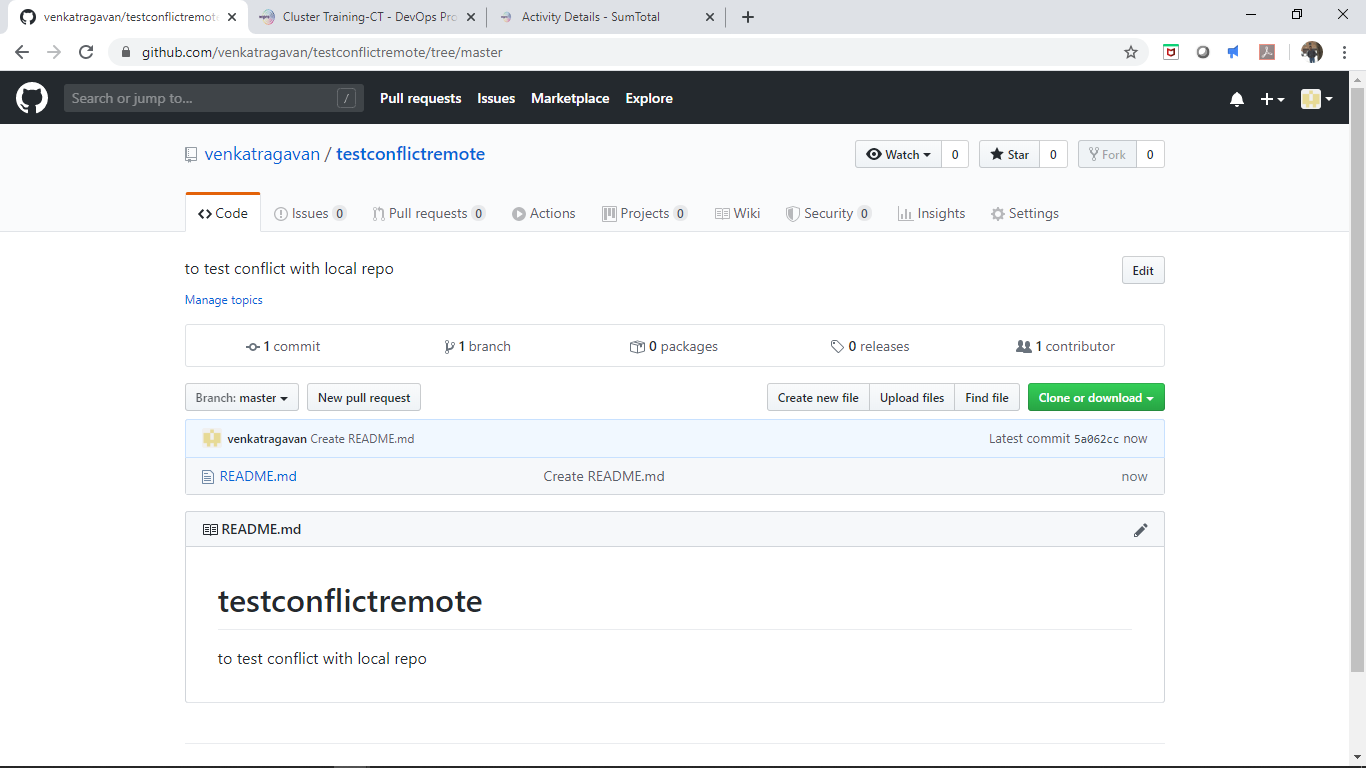




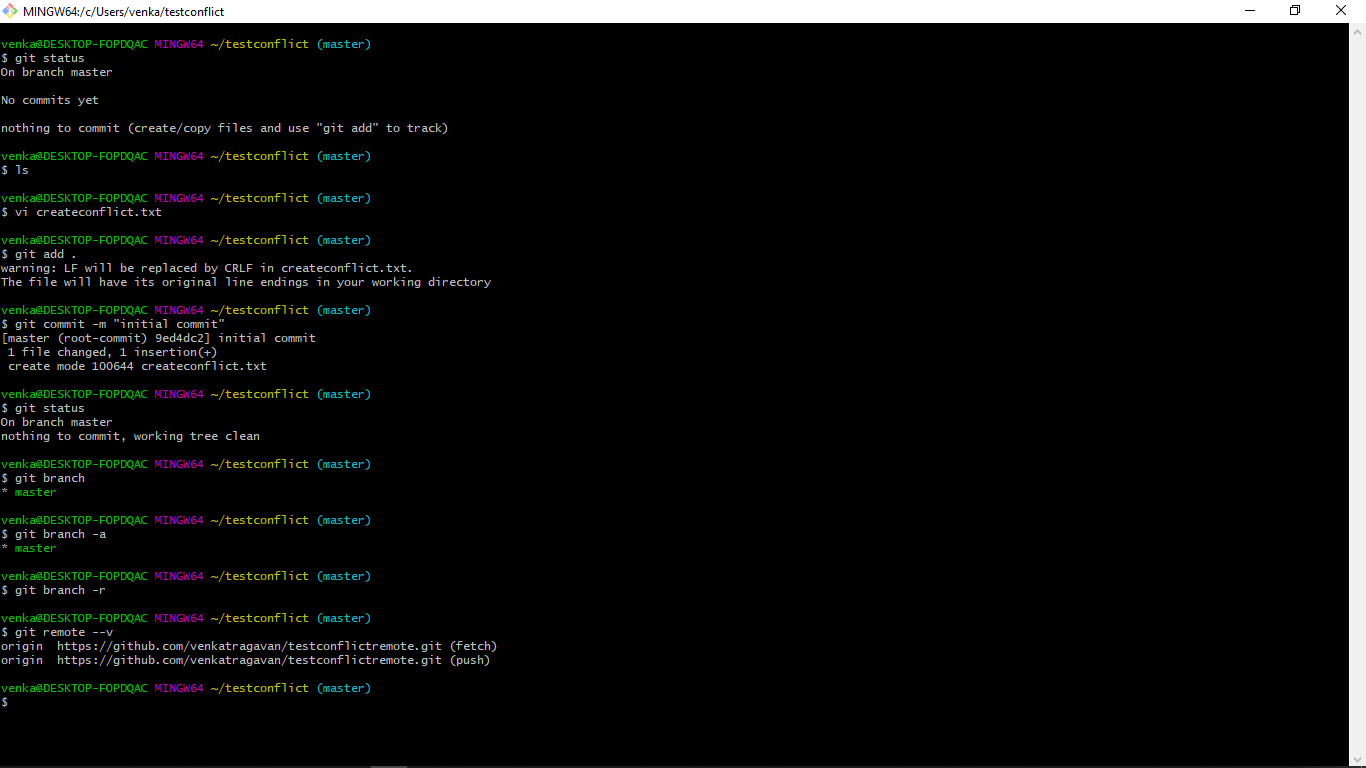
We can see a copy is created in the new directory:



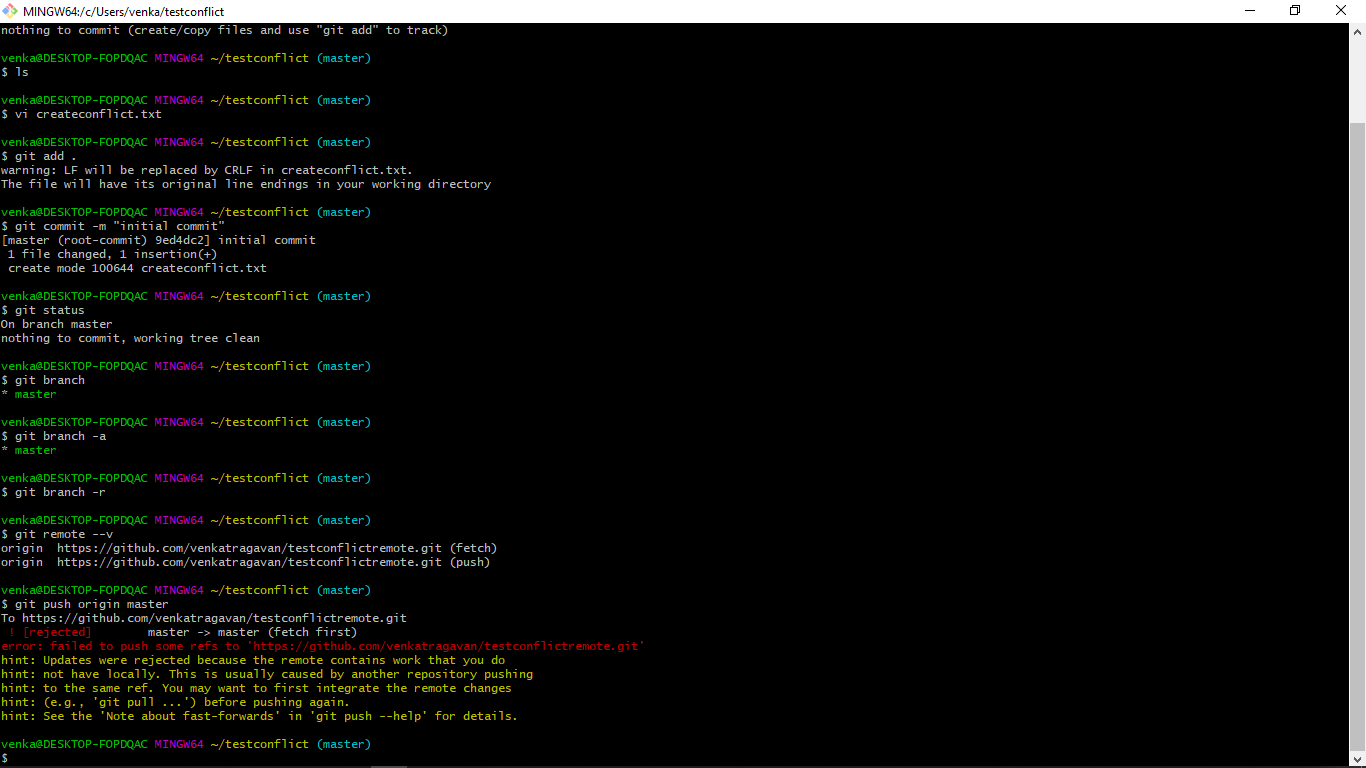




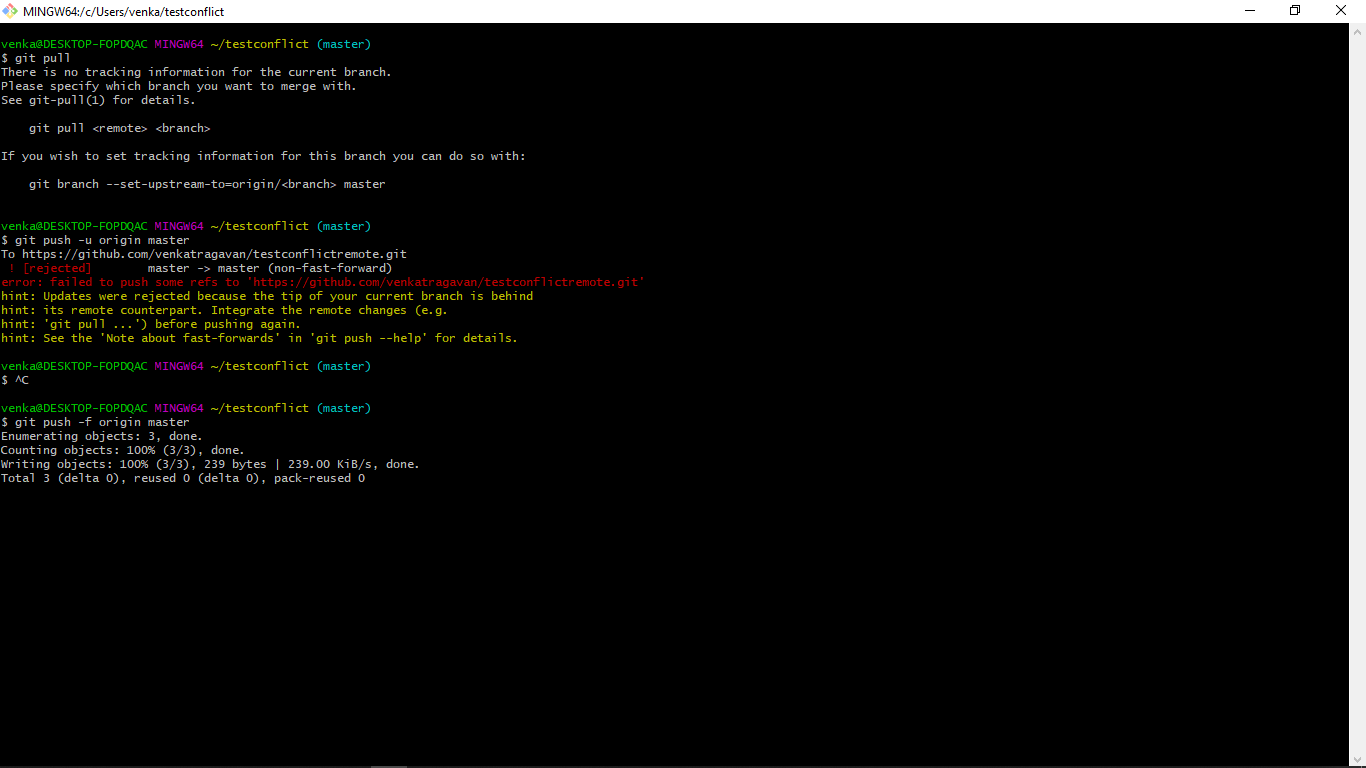




Git push failed as the remote repo is not in synch due to the README file creatd in remote repo.



Will do a force push to remote from local repo:



We can see the remote repo has changes pushed from local repo:

