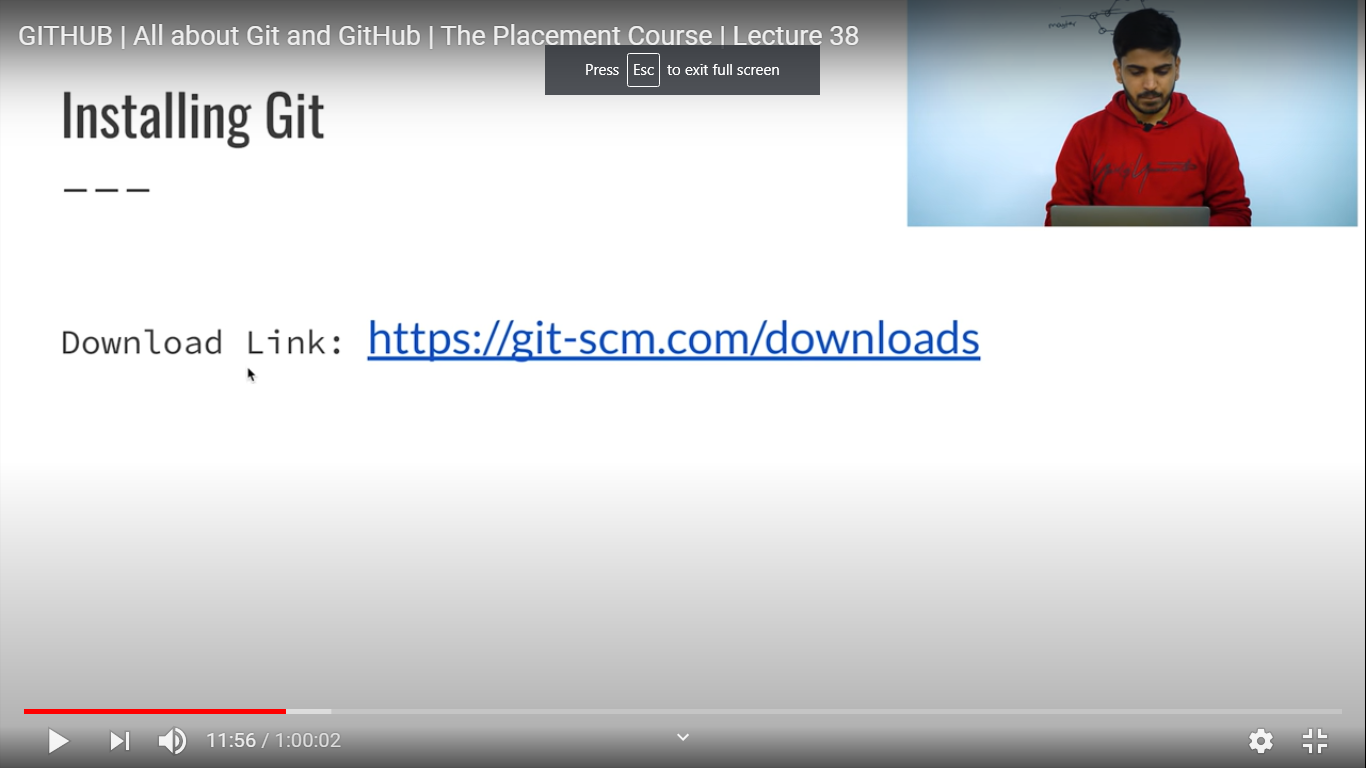
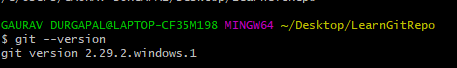


Github is cloud

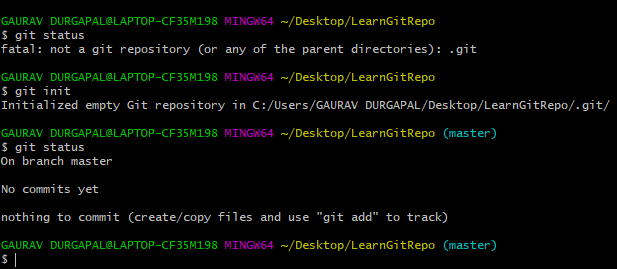


Commands:

1. git --version

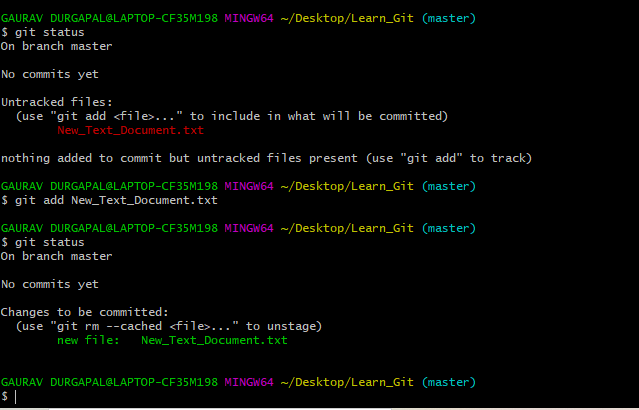


2. git init - initialize a git repository for a new or existing project

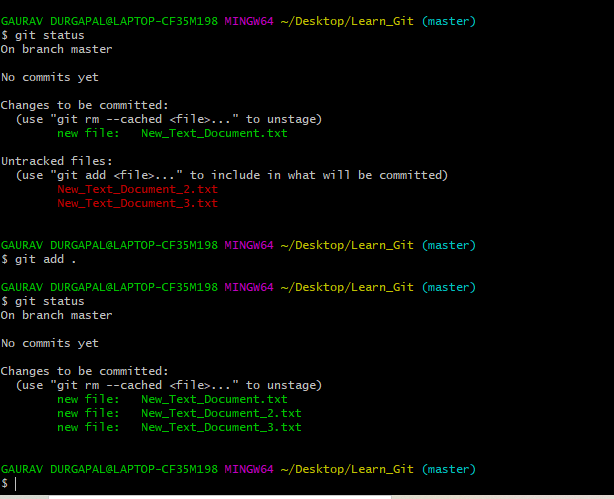


3.(a). git add :- Add one or more files from untracked area to staging area(index)



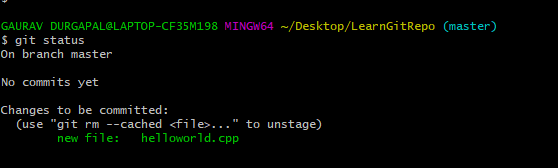


(b). git add . - Show messages for all files in which we did the changes or added

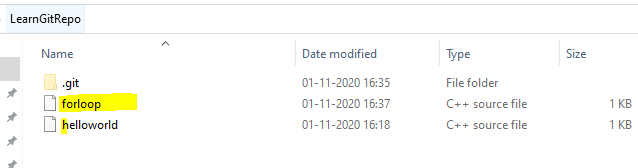


4. git status: shows the status of changes as untracked, modified, or staged.

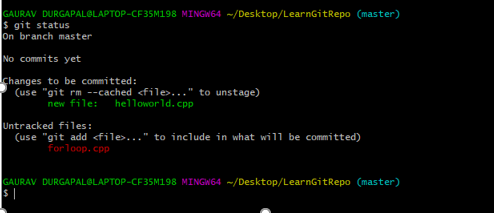
(a). File is in the staged area



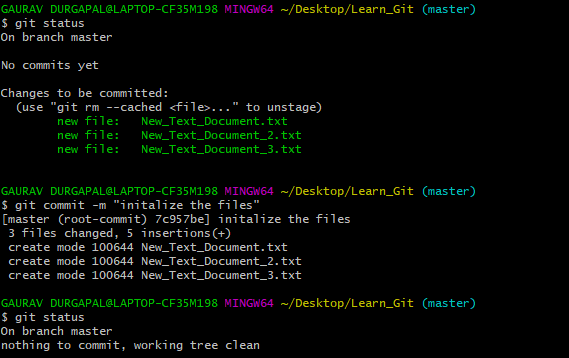
(b). Make other file forloop in LearnGitRepo

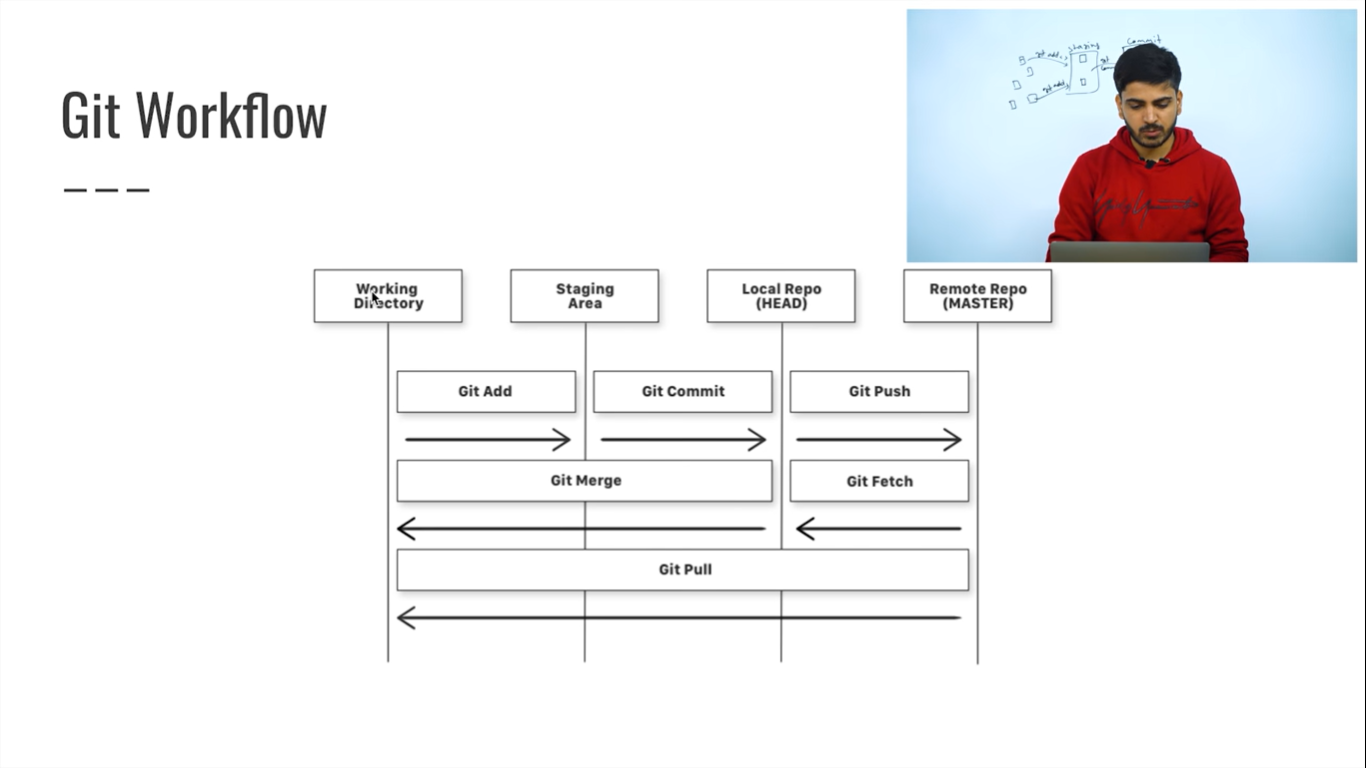


File forloop.cpp is in the untracked area



5. git commit: commit all our changes and made a checkpoint in the local repository.





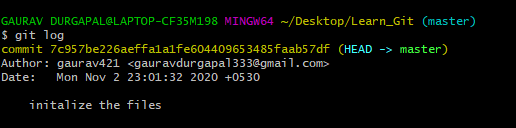
**HEAD**: the current commit your repo is on. Most of the time HEAD points to the latest commit in your current branch, but that doesn't have to be the case. HEAD really just means "**what is my repo currently pointing at**".

In the event that the commit HEAD refers to is not the tip of any branch, this is called a "detached head".  
The simple answer is that **HEAD** is a pointer/label to the most recent commit of the branch you are currently on

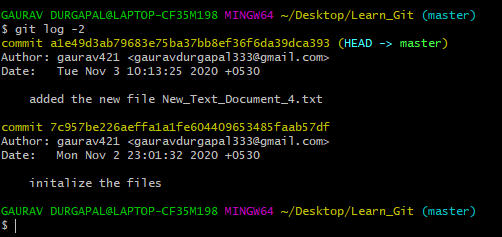
**master**: the name of the default branch that git creates for you when first creating a repo. In most cases, "master" means "the main branch". Most shops have everyone pushing to master, and master is considered the definitive view of the repo. But it's also common for release branches to be made off of master for releasing. Your local repo has its own master branch, that almost always follows the master of a remote repo.

**origin**: the default name that git gives to your main remote repo. Your box has its own repo, and you most likely push out to some remote repo that you and all your coworkers push to. That remote repo is almost always called origin, but it doesn't have to be.

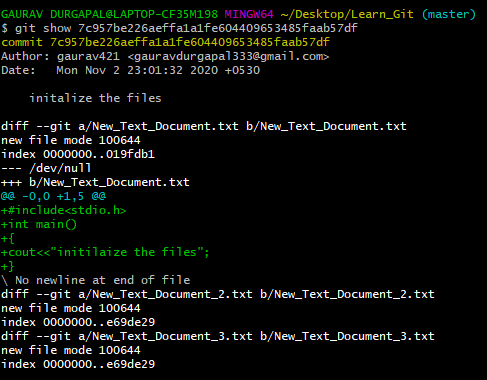
6. git log (show all commit details)



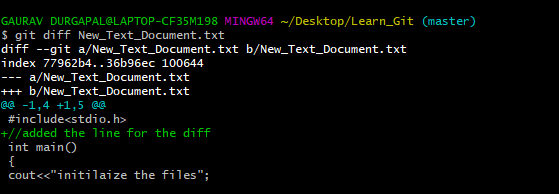
git log -n (show the last n commits)



7. git show: shows all the changes done in the commit hash



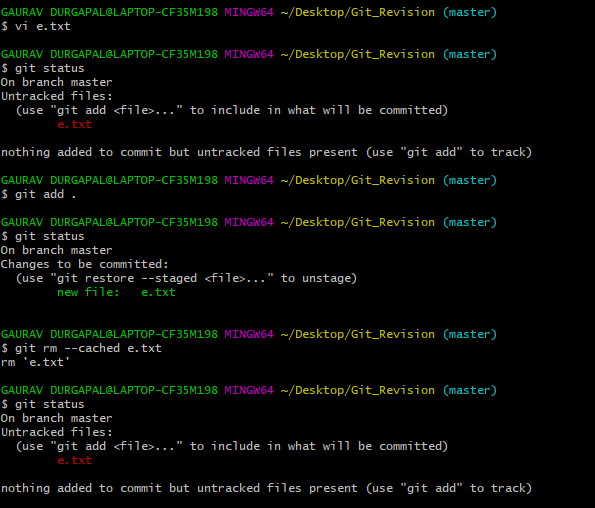
8. git diff: show diff in the file // we can see the diff if we did not commit the code changes



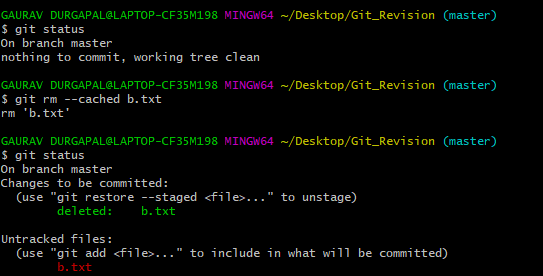
If the changes were committed, then git diff is of no use

9. git rm --cached <file\_name> (To remove files from staging area(Unstage) or If you've already committed a bunch of unwanted files, you can unstage them and tell git to mark them as deleted (without actually deleting them) with)

(a). files moved to the tracked area , now untrack them



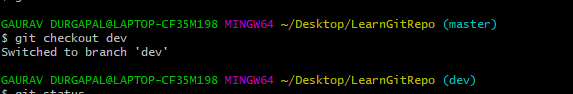
(b). for committed code



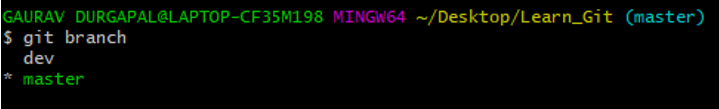
10. git branch branch\_name: To create a new Branch with name as branch\_name



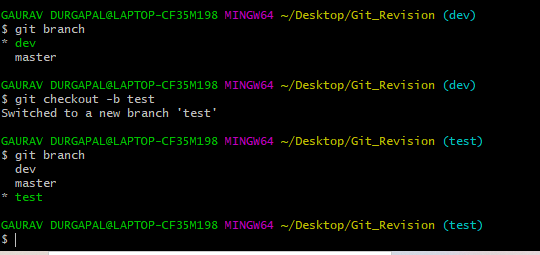
11. git checkout <branch\_name>: To switch from current branch to another

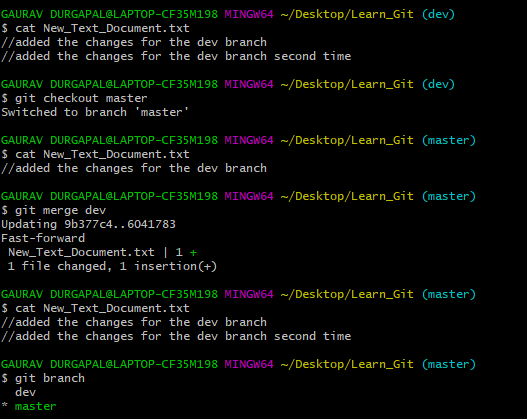


12. git branch:- tell in which branch currently we have

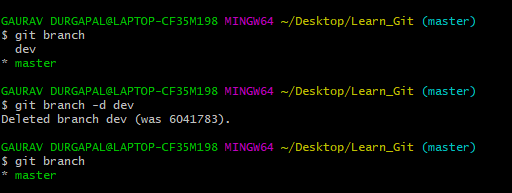


13. git checkout -b: create and switch to the branch

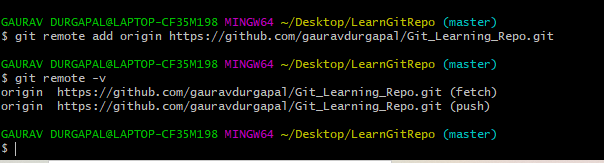


14. git merge <branch\_name> // To merge a branch into current branch  
Make sure all changes are commit when you create new branch and when you switch from dev to master for merging. After merging, no need to commit  


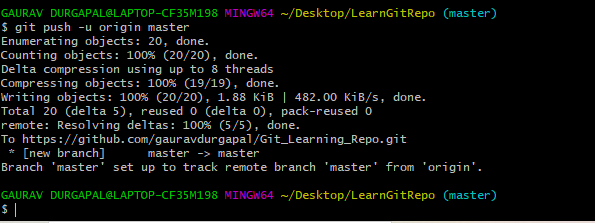
15. git branch -d branch\_name //used to delete the branch



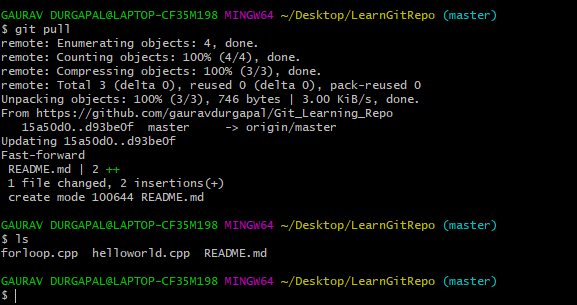
16 . git remote add origin url // add the new remote



17. git push // updates the remote repository with any commits made locally to a branch.

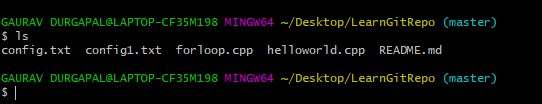


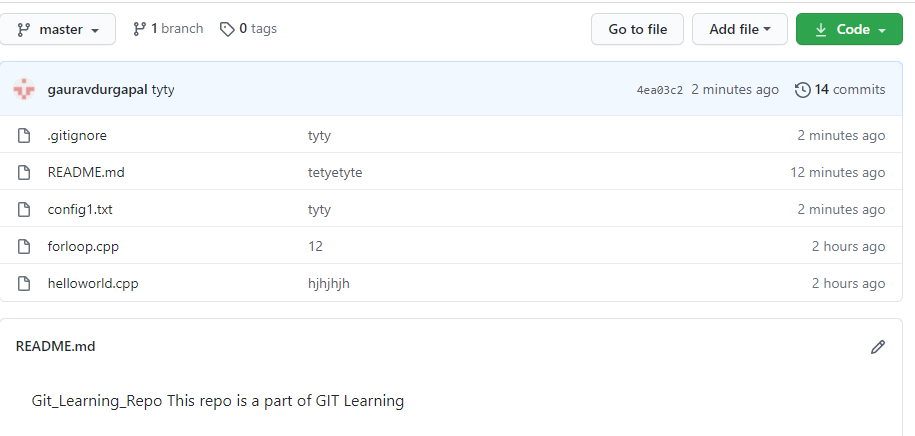
18. git pull (updates the local line of development with updates from its remote counterpart. Developers use this command if a teammate has made commits to a branch on a remote, and they would like to reflect those changes in their local environment.

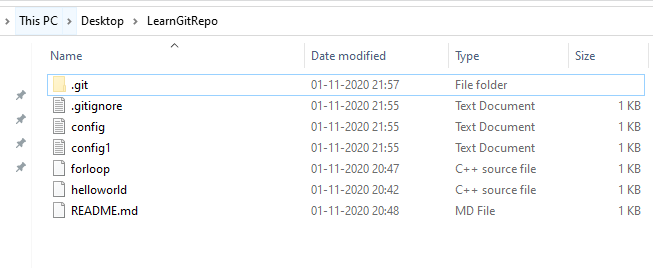


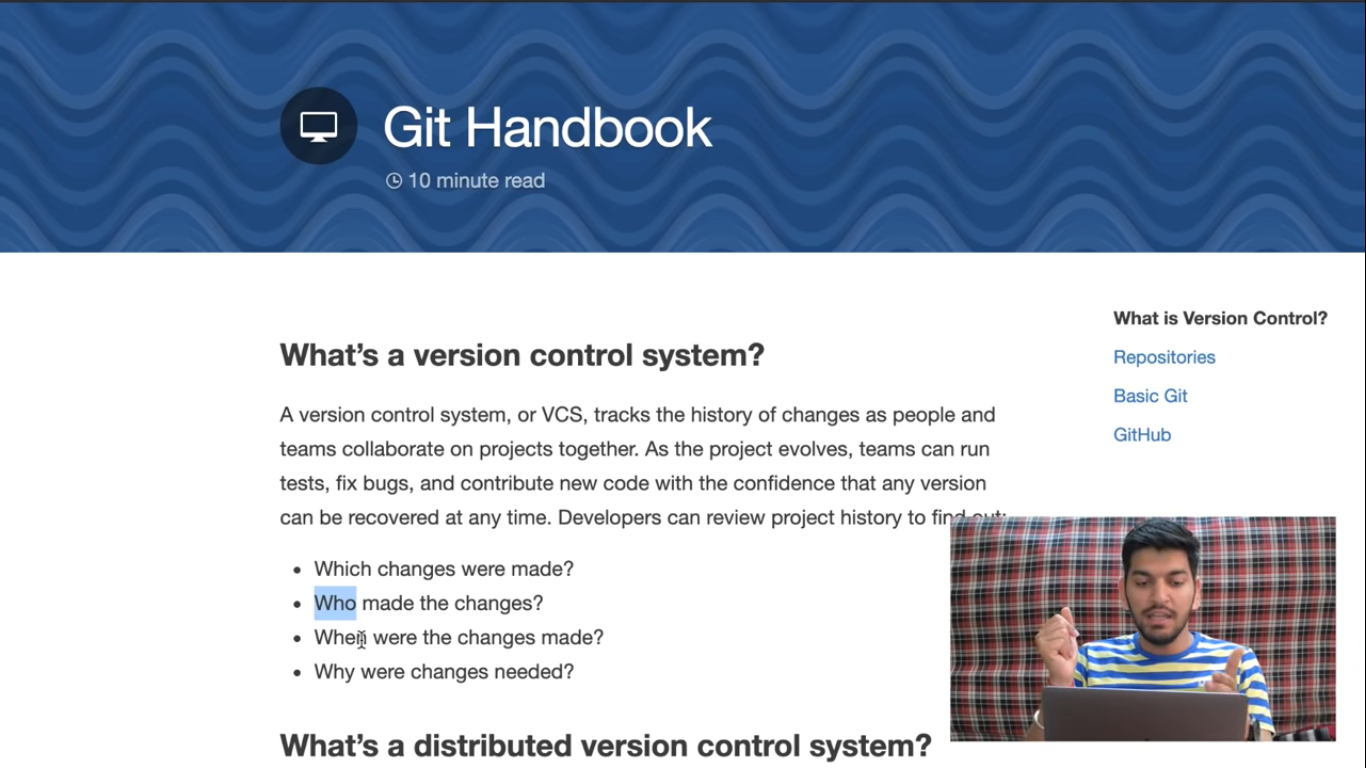
19. gitignore



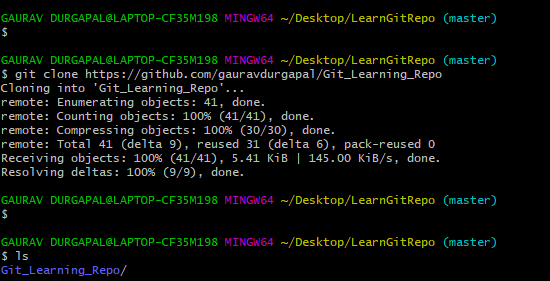




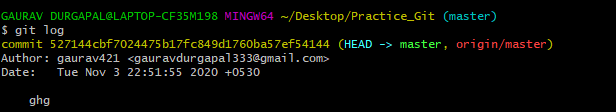


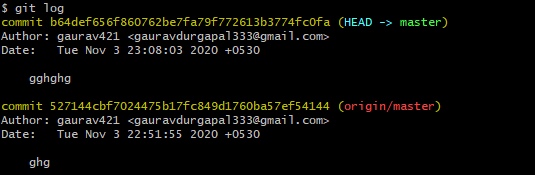


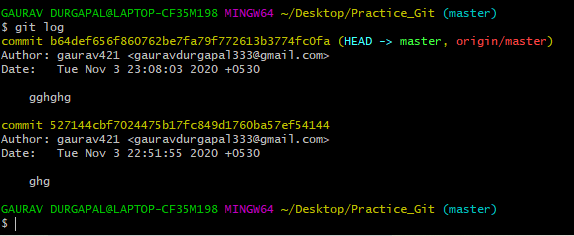
20. Git clone:- git clone creates a local copy of a project that already exists remotely. The clone includes all the project’s files, history, and branches.



Your branch is up to date with origin/master. It means remote files and local files are equal

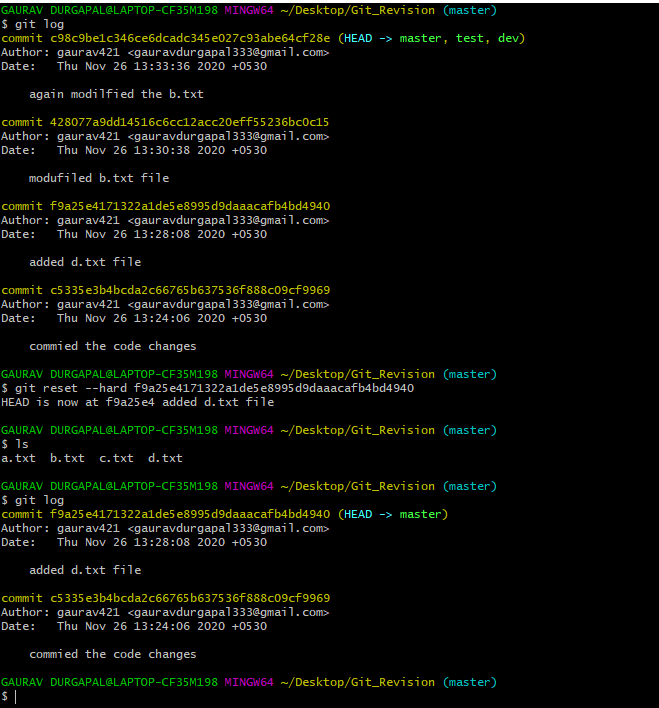




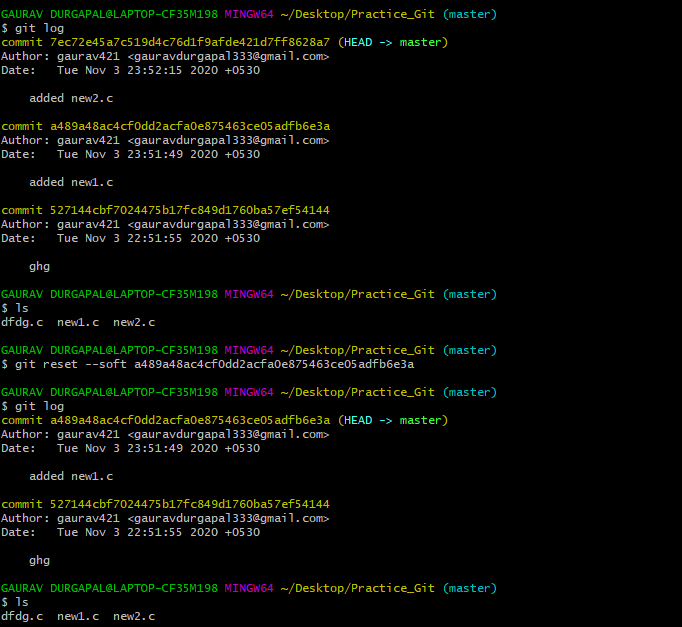


21. git reset –hard //remove commits and code changes

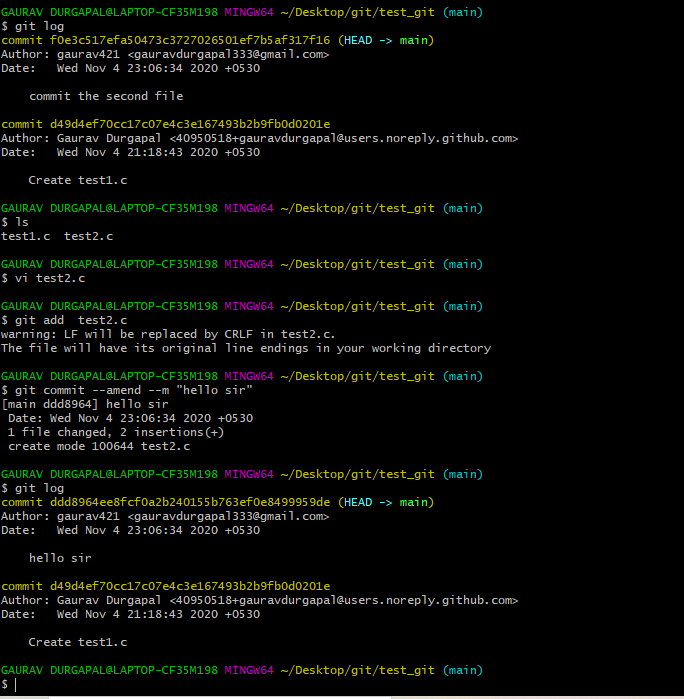




22. git reset –soft //remove the commits but changes of latest commits are there



23. git commit –amend : git commit --amend command is a convenient way to modify the most recent commit. It lets you combine staged changes with the previous commit instead of creating an entirely new commit



24 . How to remove the merge conflicts

https://www.youtube.com/watch?v=JtIX3HJKwfo

25. Forking and Cloning

<https://www.toolsqa.com/git/difference-between-git-clone-and-git-fork>

26. Remove the remote origin