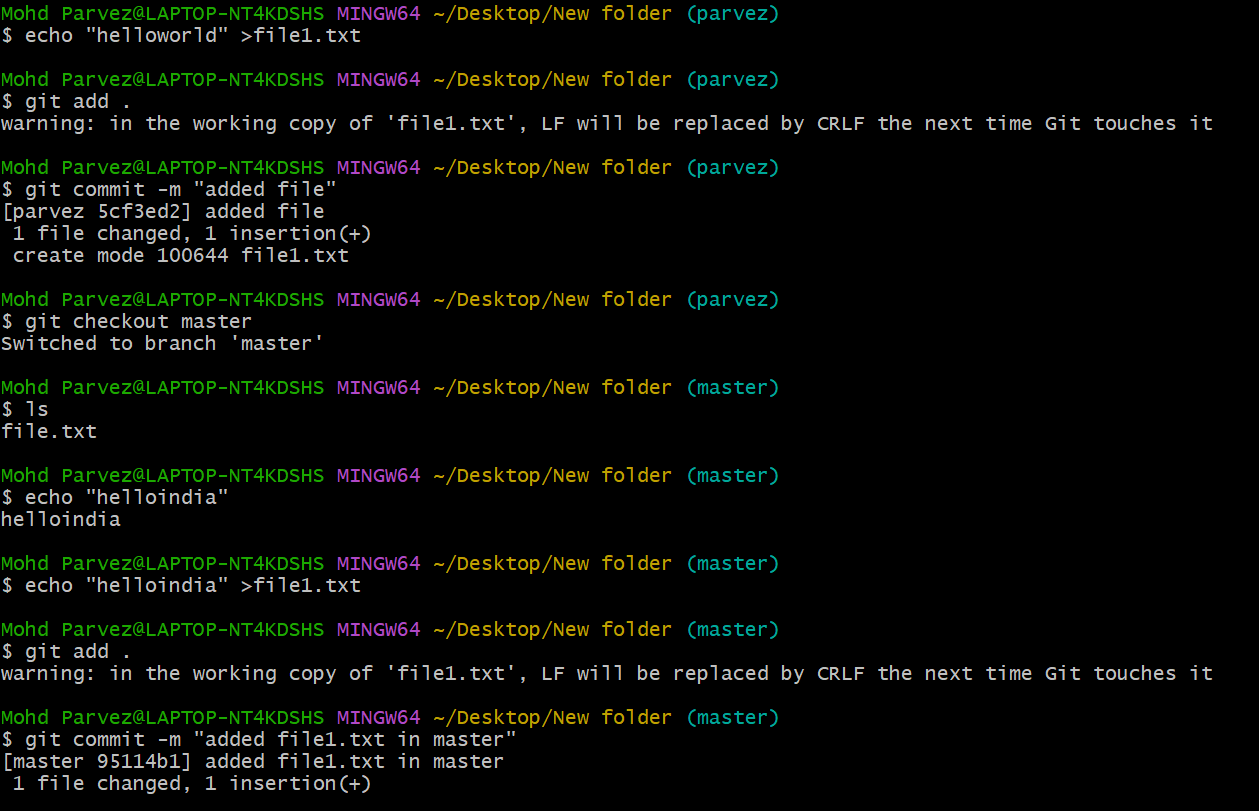
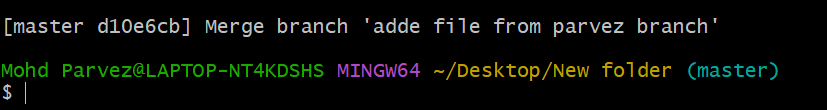
GIT AND GIT HUB CALLENGES

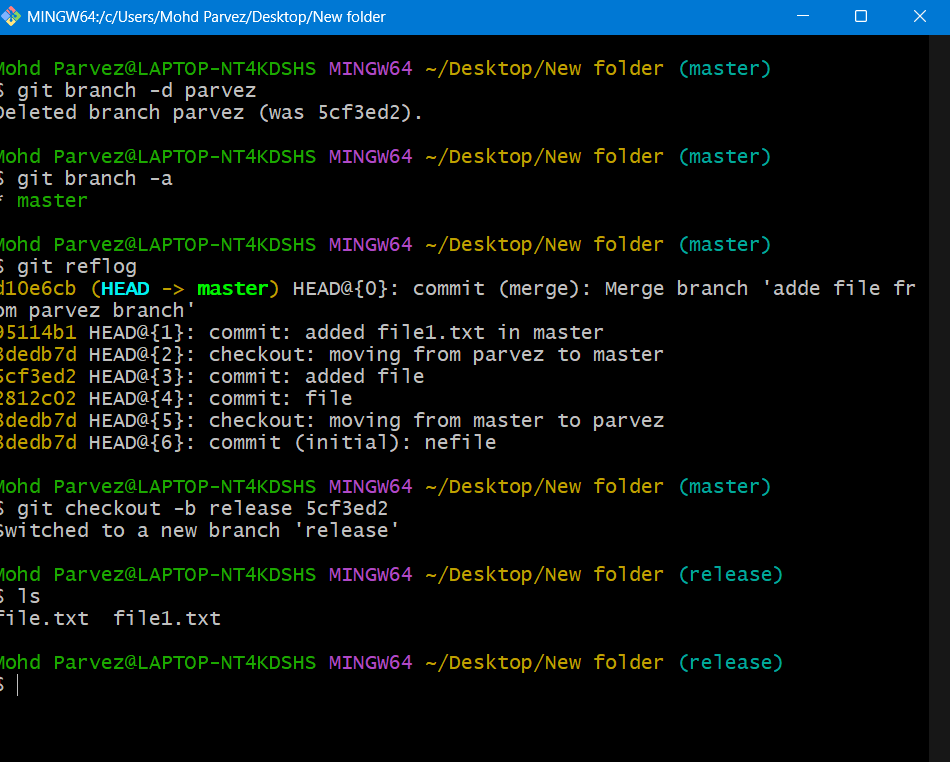
1. **Resolve Merge Conflicts**
   * Create a merge conflict intentionally (two users editing the same line).
   * Resolve the conflict and push the changes.





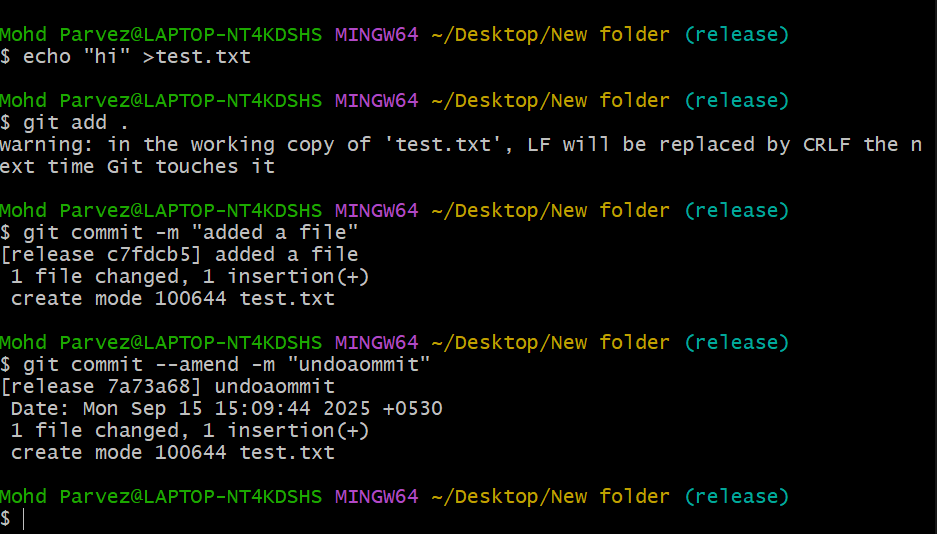
* Create the file in your repo and create in another user with the same name of the file
* Then merge the repo from one to another repo user the merge conflict occurs because of the same file name and do vi to that file and keep the content you want and save it and add and commit and merge .

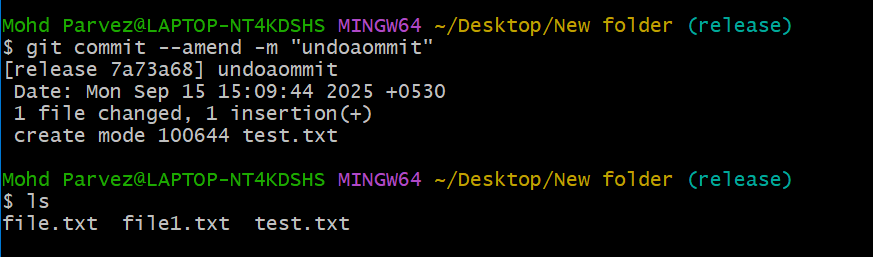
1. **Recover Deleted Branch**
   * Delete a local branch and then recover it using the reflog.



* Git branch -d <branch name> : to delete the branch
* Git log : check the log
* Git checkout -b release <commitid> : the commit id of deleted branch it will recover the deleted branch

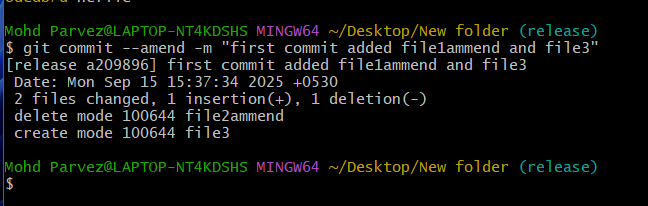
1. **Undo Wrong Push**
   * Push a wrong commit to GitHub, then undo it without losing history.



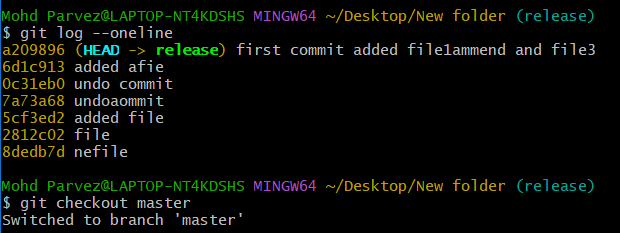
* . Identify the bad commit
* Use to find the commit hash you want to undo. 
* Revert the commit
* This creates a new commit that undoes the changes
* . git revert <commitid>: This creates a new commit that undoes the changes.
* Now push the new "undo" commit to your remote branch.: git push origin <branch-name>

1. **Amend a Commit**
   * Make a commit, then add a missing file to it using git commit --amend.

git commit --amend: Modifies the last commit (can change

the commit message or add changes). 

1. **Cherry-pick a Commit**
   * Take a specific commit from one branch and apply it to another branch.

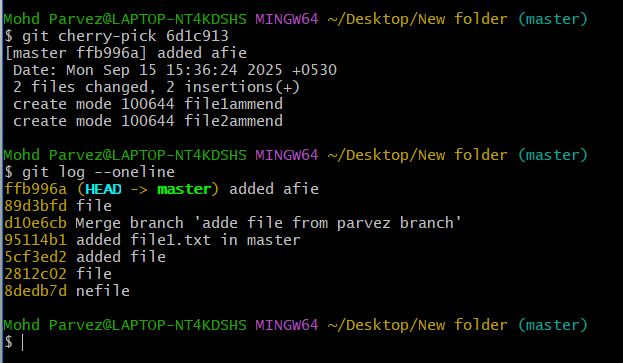


git log --oneline (to check the commit id and copy commit id)

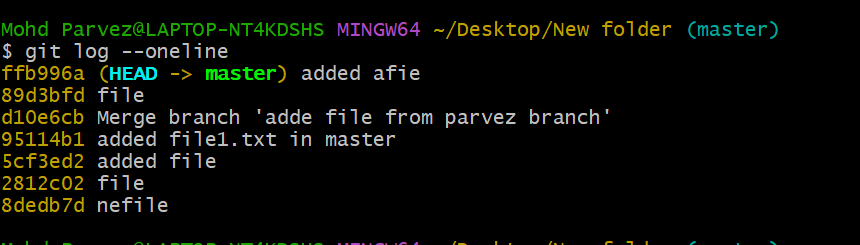
⦁ switch to other branch and enter

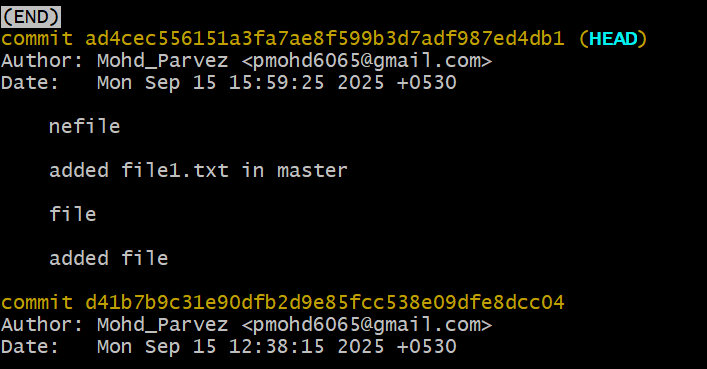
⦁ git cherry-pick (commit \_id)

⦁ git log --oneline (hence you can get the spicifc id in other branch)

****

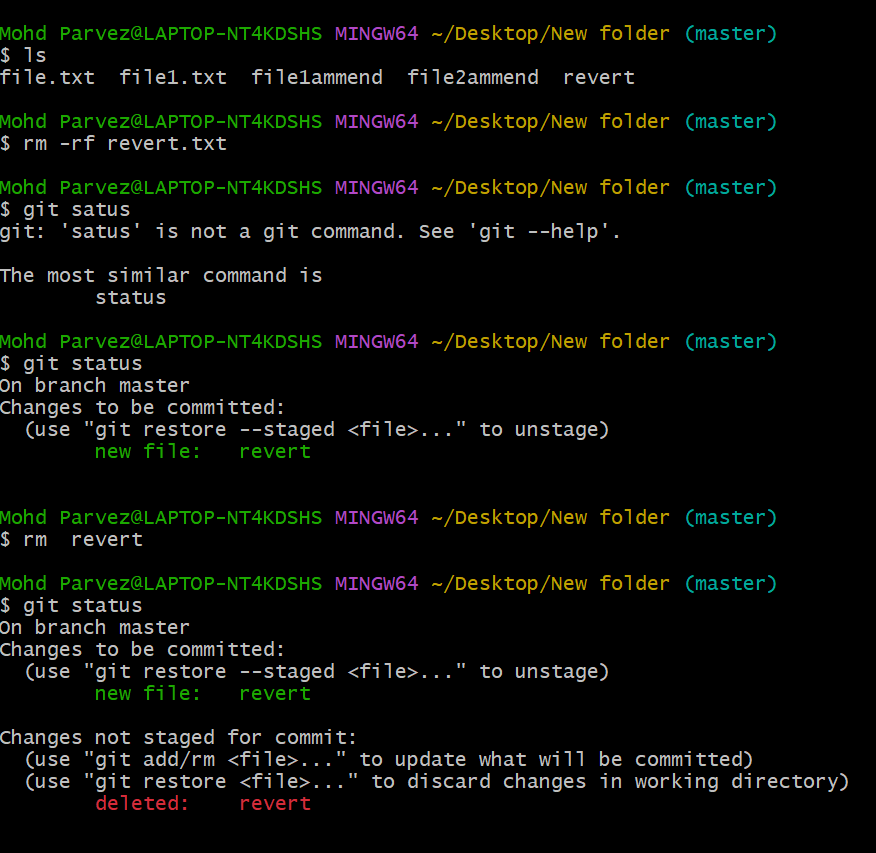
1. **Interactive Rebase**
   * Reorder and squash multiple commits into a single clean commit.

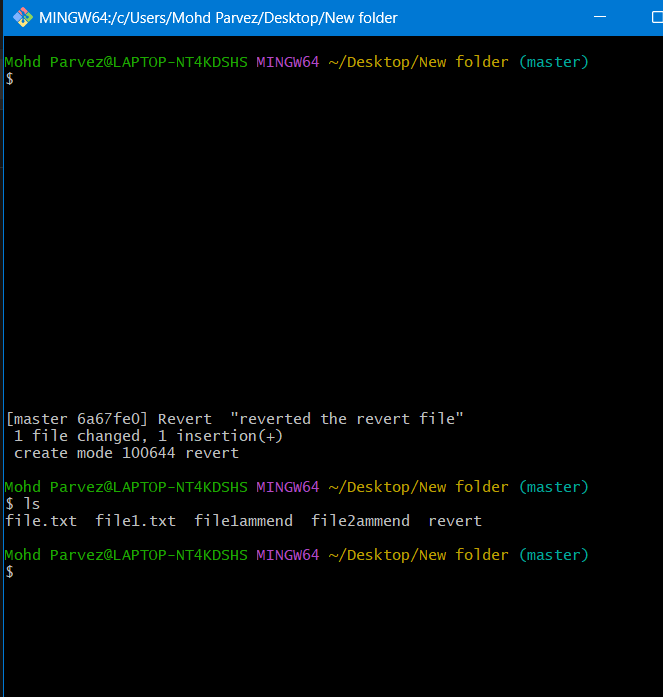




git log --oneline git rebase -i --root a vi editor will open there you need to change the commit id (pick to squash) except the one you want to add in again check all log using git log --oneline

1. **Tagging & Release**
   * Create a version tag (v1.0), push it to GitHub, then delete and restore it.
2. **Clone with Sparse Checkout**
   * Clone only a subdirectory of a repo using sparse checkout.
3. **Reset vs Revert Challenge**
   * Demonstrate the difference between git reset --hard and git revert in a repo.





**Git rebase –hard**

⦁ git log --oneline

⦁ git reset --hard commit\_id (to delete a commit from logs without keeping history)

⦁ after using these command you can't find the commit in git log or git log --oneline

⦁ to get the commit id we need to use

⦁ git reflog

⦁ git reset --hard commit\_id (again it reset the changes till that commit id by deleting all

the next chenges after that commit)

⦁ git reset --soft (it will remove the commit id and move the file to untracking area)

⦁ git reset commit \_id (it will remove the commit id and move the file to tracking area

again we need to commit it)

⦁ main differece of git reset and git revert is

git reset : it wil reset the changes till a specific commit by removing the hsitory its not

safe to use)

⦁ git revert : it will modify the commit by keeping the history and creating one more

commit for it)

git log --oneline

⦁ git checkout commit \_id ( it will show the head at commit\_id other are detached as

wll it will show a commit id in place of branch)

⦁ git checkout -b commit\_id (it will create a branch till that commits without copying all

content from it )

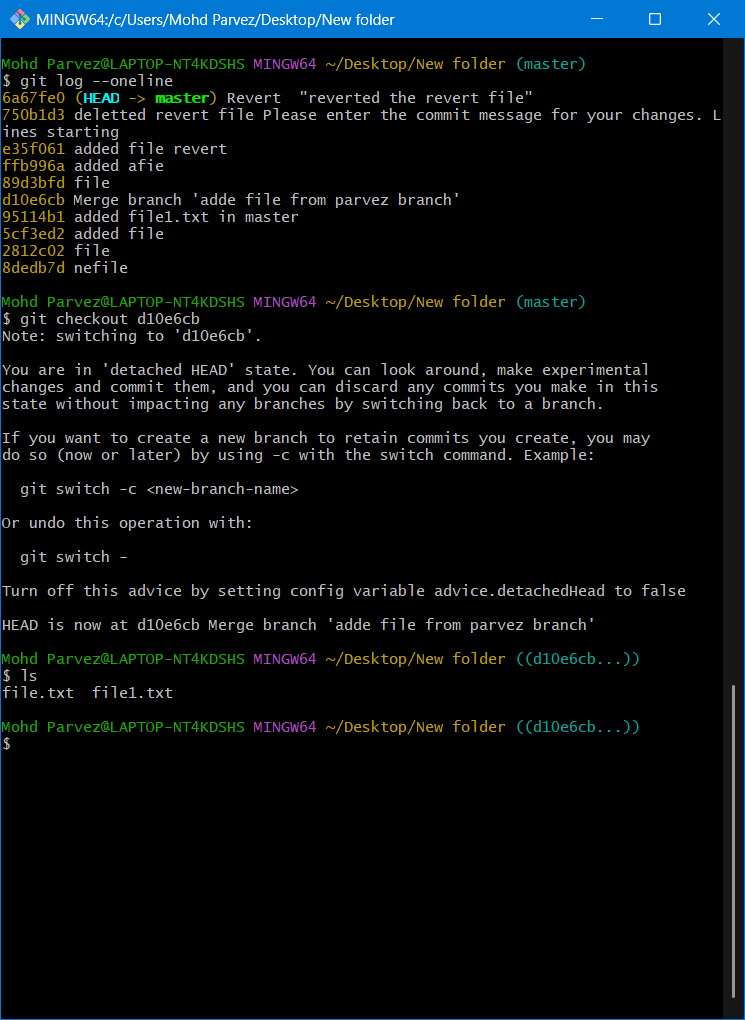
⦁ ls (we check all changes updated till the given commit\_id)

⦁ git log --oneline (it will show logs only before our given commit\_id)

git checkout commit\_id command enables us to create a branch till the specific commit

without cloning the whole branch..

1. **Detached HEAD Challenge**
   * Checkout a specific commit (detached HEAD state) and create a new branch from it.



git log --oneline

⦁ git checkout commit \_id ( it will show the head at commit\_id other are detached as

wll it will show a commit id in place of branch)

⦁ git checkout -b commit\_id (it will create a branch till that commits without copying all

content from it )

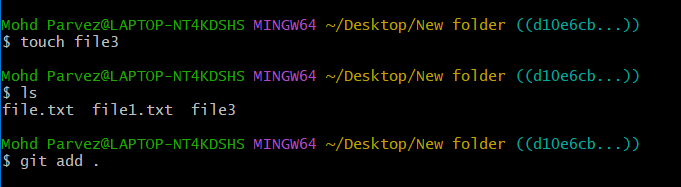
⦁ ls (we check all changes updated till the given commit\_id)

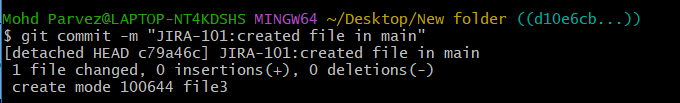
⦁ git log --oneline (it will show logs only before our given commit\_id)

git checkout commit\_id command enables us to create a branch till the specific commit

without cloning the whole branch..

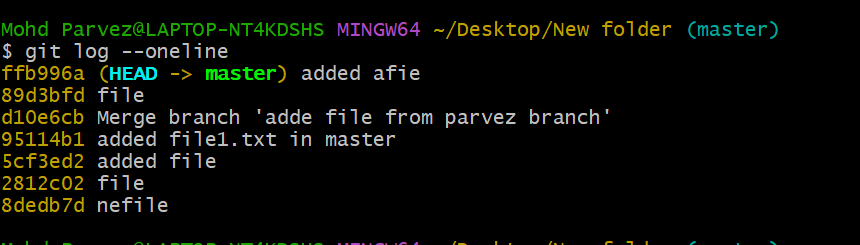
1. **Git Hooks Challenge**
   * Configure a pre-commit hook to reject commits without a message format (e.g., must start with JIRA-XXX).

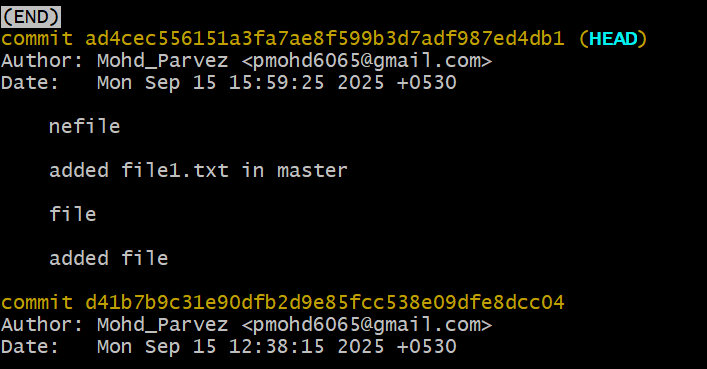




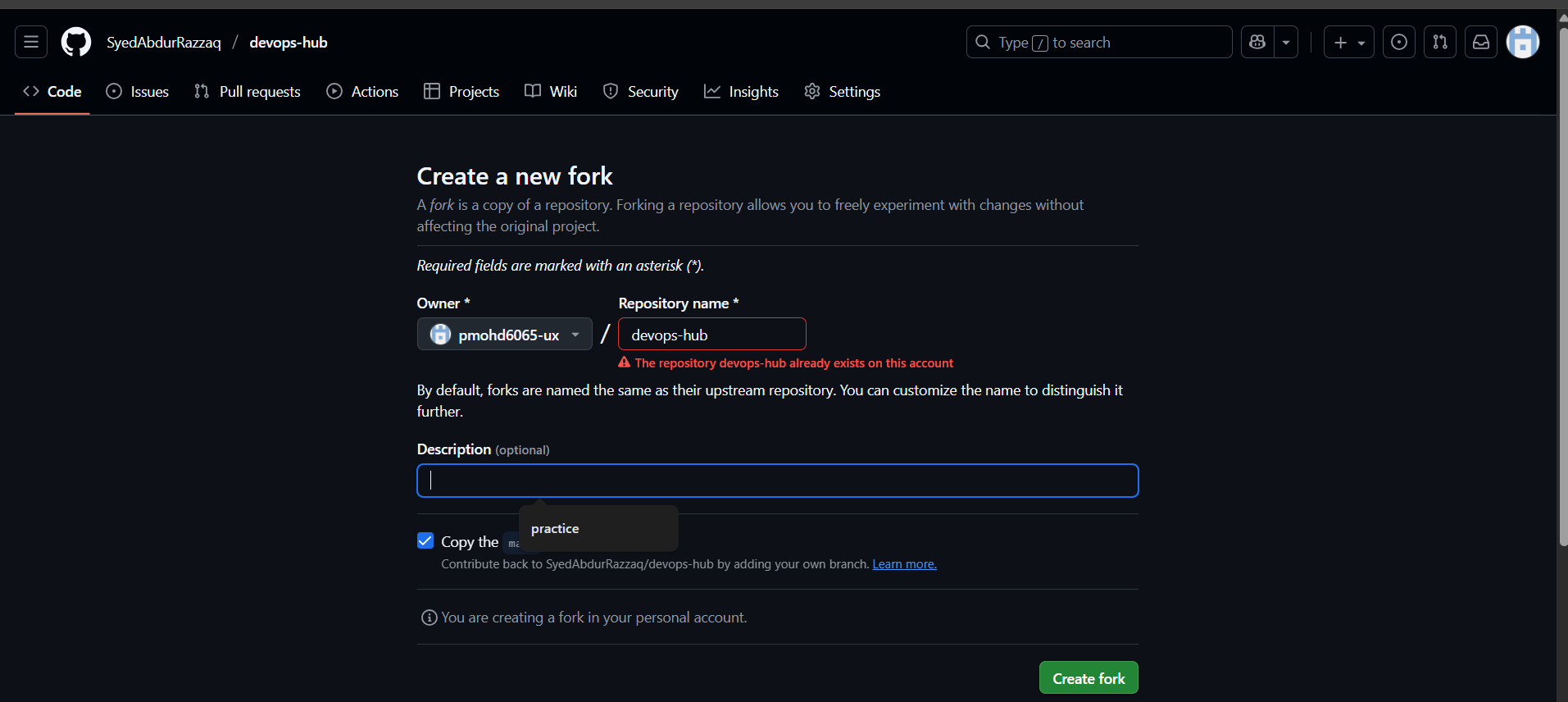
1. **Squash Merge vs Rebase Merge**
   * Show the difference between squash merge and rebase merge with evidence.

Squash merge:



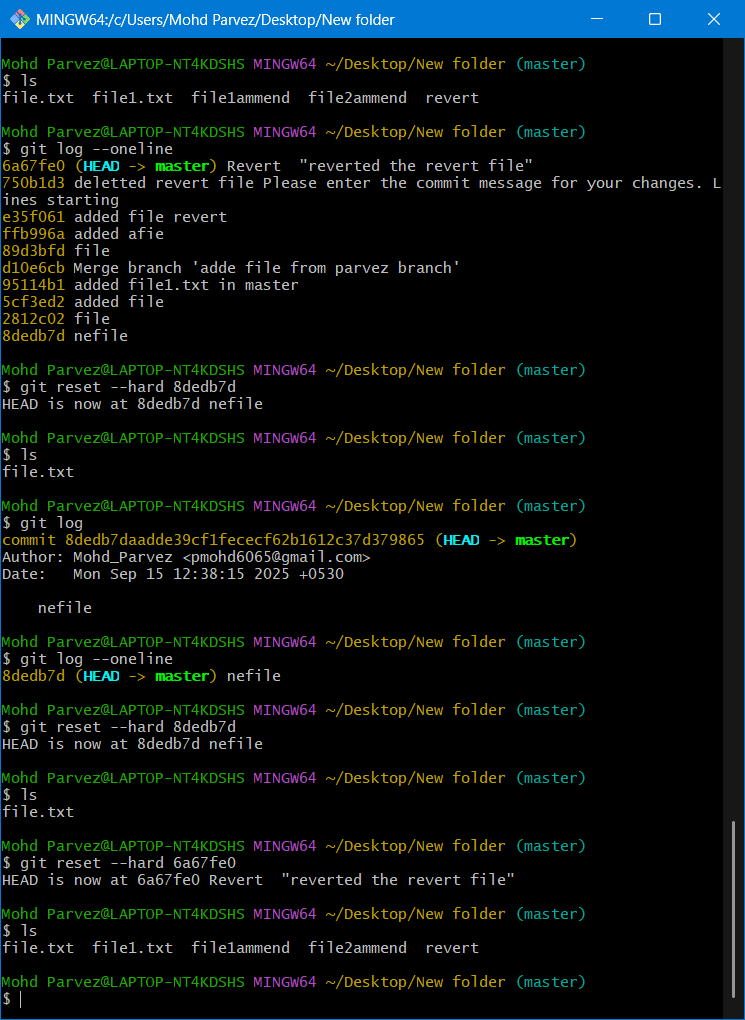


1. **Fork & Pull Request Workflow**
   * Fork a repo, make a change, and submit a pull request to the original repo.



to make a fork request you need to add the collaborators ⦁ then click on fork on right side of your github account and add any repo of your collaborator acoount which is public.If it's private then we need to make a request and he needs to accept the request for the repo under fork requests. ⦁ fork request helps in merging one repo from one account to another account . 19 ⦁ pull request helps in merging the branches on git hub

1. **Recover Lost Commit**
   * Commit something, reset hard, and then recover it using git reflog.



ls (check for all files)

⦁ git log --oneline (check all logs and copy the commit id till where you want to reset

the account)

⦁ git reset --hard commit\_id (paste the copy id till where you need to remove the

commits and updates)

⦁ ls (check for list of files you will find all files has been deleted till the commit id)

⦁ git log --oneline (check for logs you will find all logs has been removed )

⦁ git reflog (check for log and copy the commit id which got removed from logs )

⦁ git reset --hard commit\_id (paste the commit id )

20

⦁ git log --oneline (now you will find all commits is recovered)

⦁ ls (finally all commits has been recovered)

⦁ git reset --hard (used to delete the commit without keeping any history of commits

if you want to recover the commit you need to use git reflog and copy the id )-------it is used

for deletion and recovery of commits

⦁ git reset --soft (it removes the commit and move the file to untracking area (you need

to add it again and needs to make new commit)

⦁ git reset commit\_id (it is mixed in nature it will remove the commit and moves the file

to tracking area hence you need to commit it back)