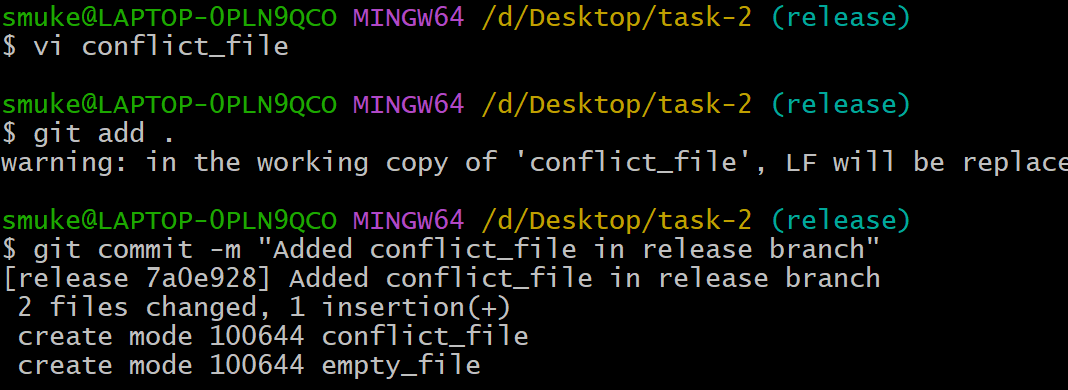
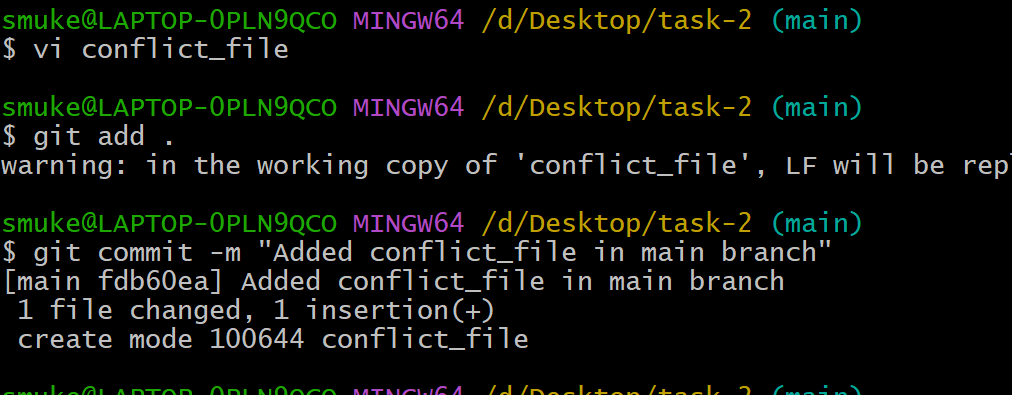
**Git Challenges**

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

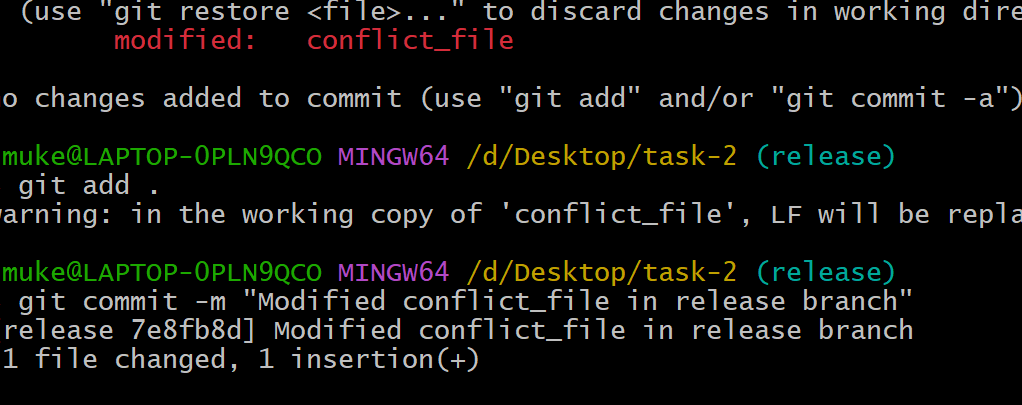
Step 1: create a file in one branch and commit it.



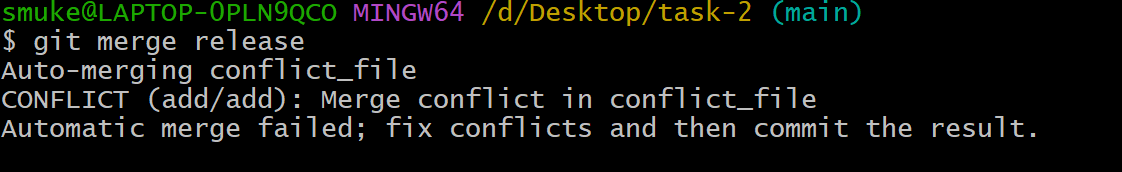
Step 2: switch to another branch and create a file with same name and commit it



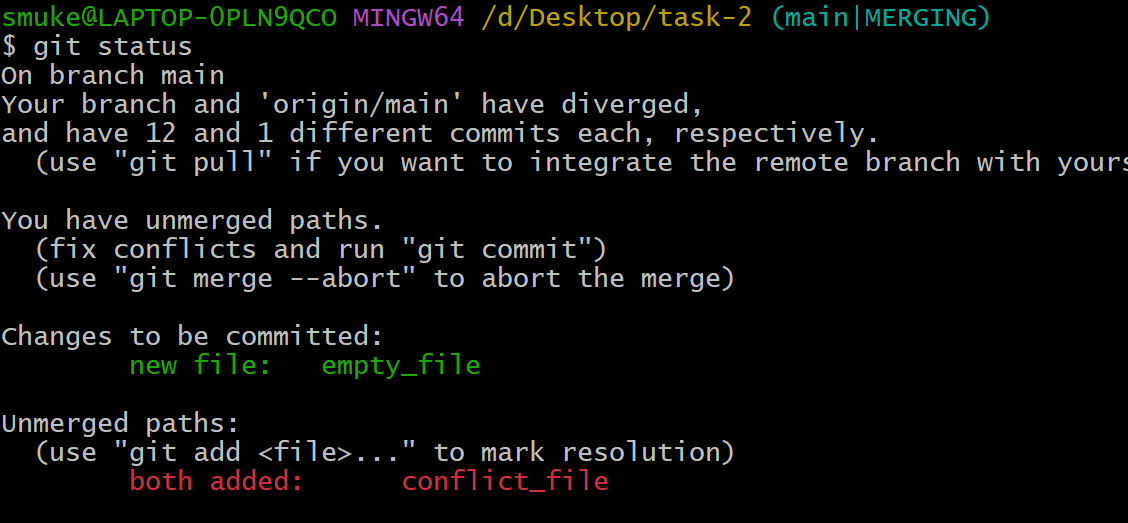
Step 3:Switch to previous branch modify the file and commit it.



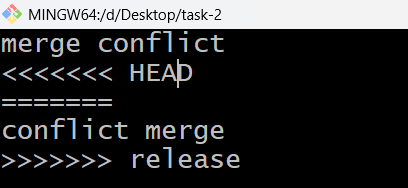
Step 4: Now comeback to other branch and give **git merge branchname c**onflict will occur

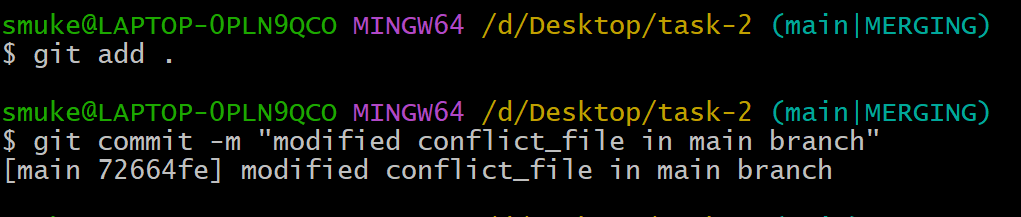


Step 5: Use **git status** to find which file has conflict issue



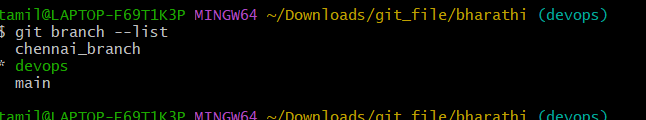
Step 6: Manually change the data of the file and commit again.



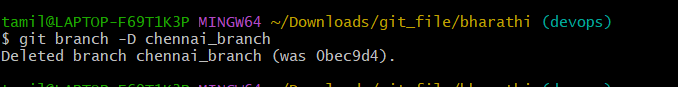


1. **Recover Deleted Branch**
   * Delete a local branch and then recover it using the reflog.

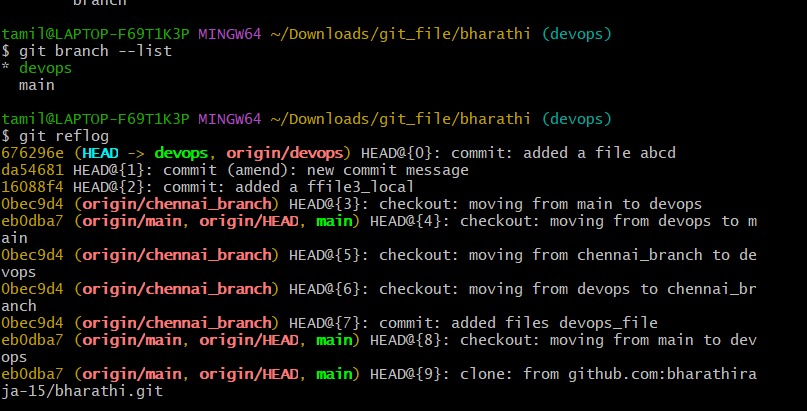
I just checked branch list how many branches were there



I delete one branch

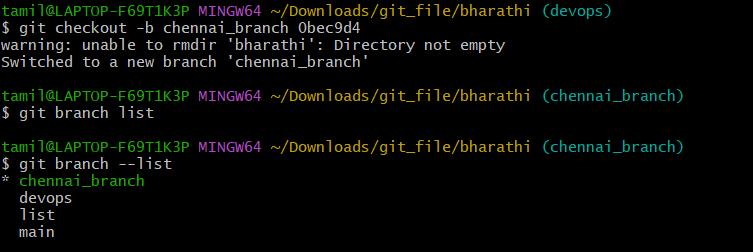


Then I check again the branch list, and there is no Chennai\_branch   
then I use it “git reflog” to check the code or ID able to see there

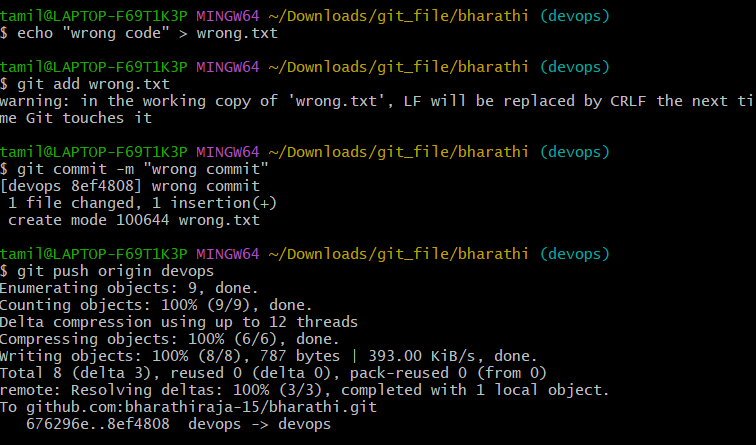


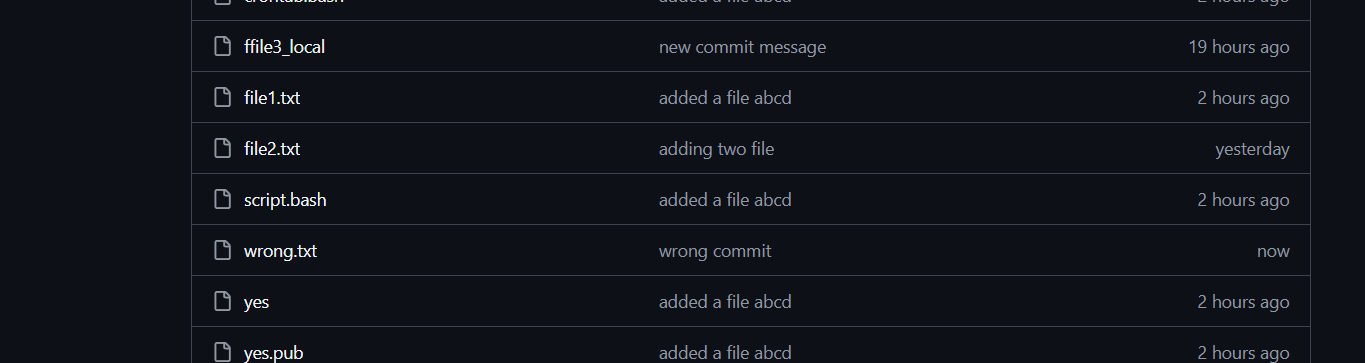
Then I use the command “ git checkout -b Chennai\_branch (id )

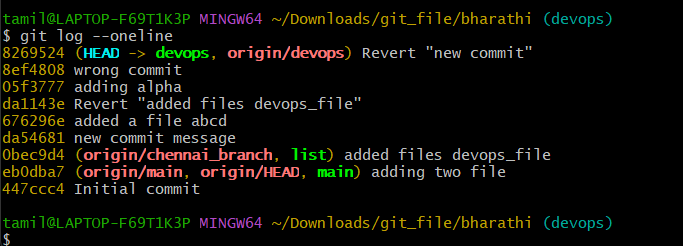
Then wit will be re



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



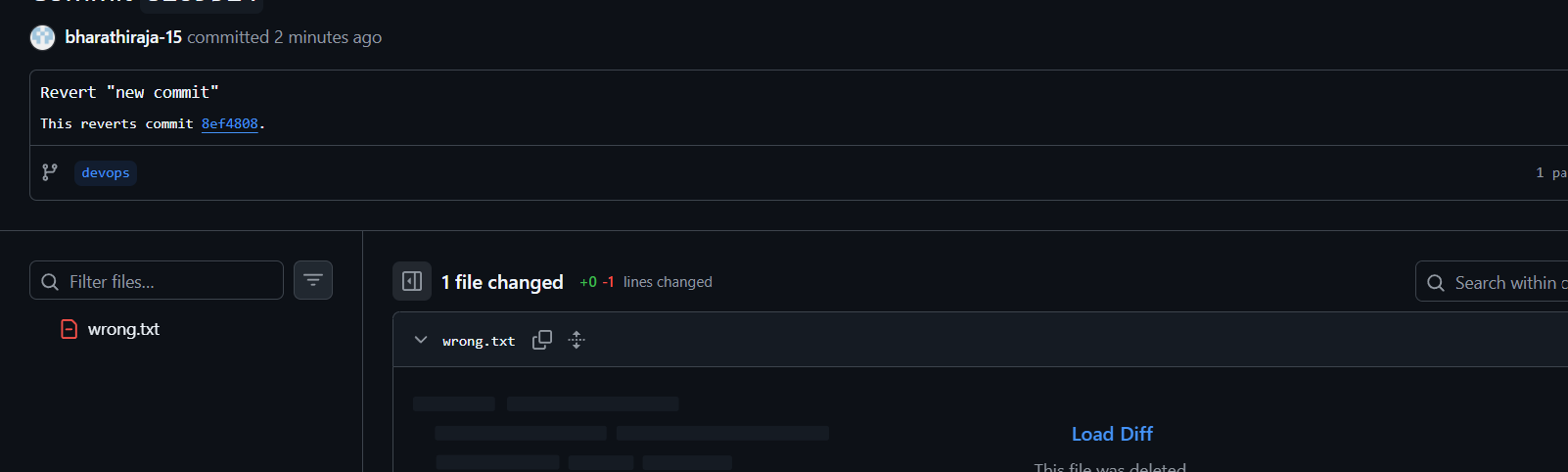




Git log –oneline, if we give it will show like that. Copy the one ID, the one we want to revert

Then “ git revert (id.no) it will show one page edit page I should write “new commit“ then

“Git Push origin devops” use this command then it will updated



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

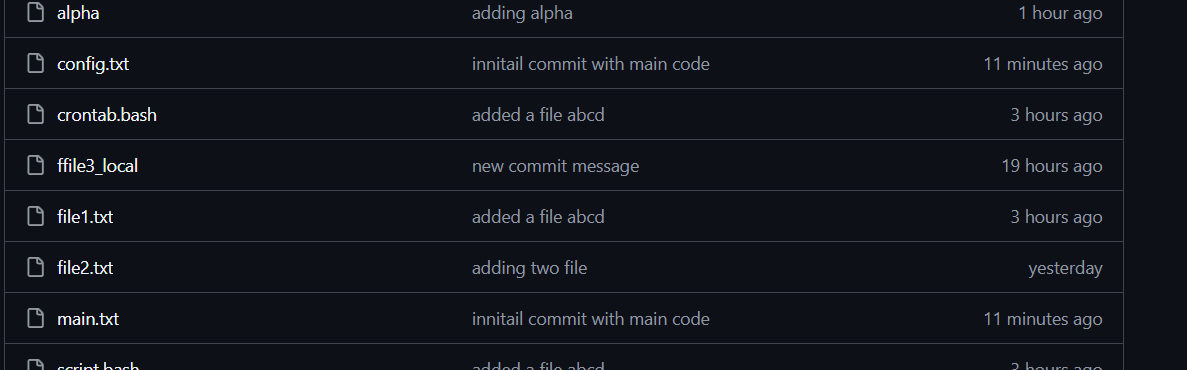
(1 file) echo "main code" > main.txt

git add main.txt

git commit -m "Initial commit with main code"  
(2nd file) echo "config=123" > config.txt

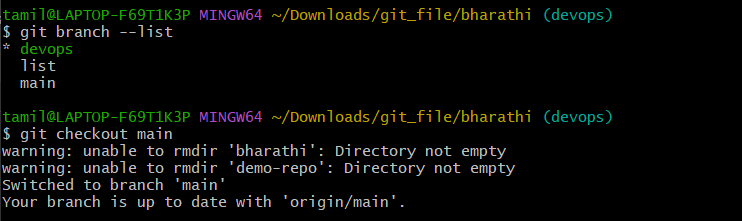
git add config.txt

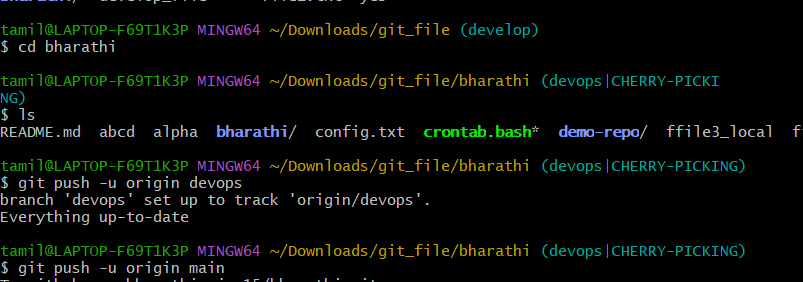
Using the command “git commit—amend” we can add both files to the commit. after that we will be able to see that both config.txt and the main. txt files are included in the same commit.



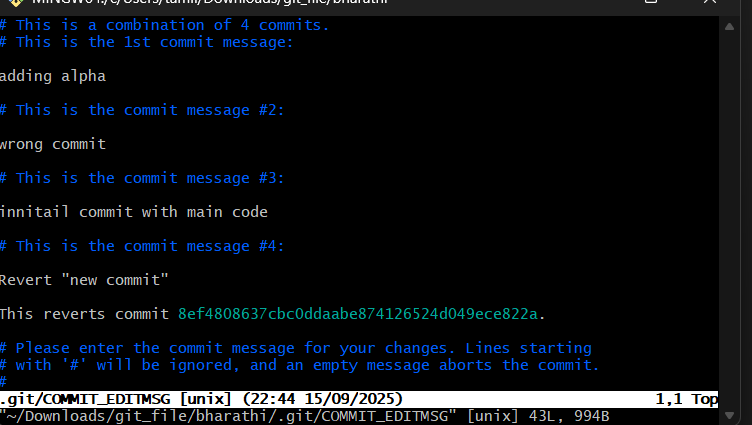
1. **Cherry-pick a Commit**
   * Take a specific commit from one branch and apply it to another branch.

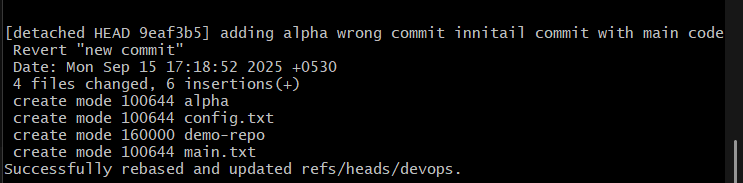
I check branch list   
I switched to main branch

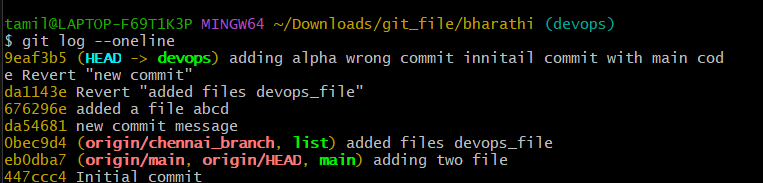


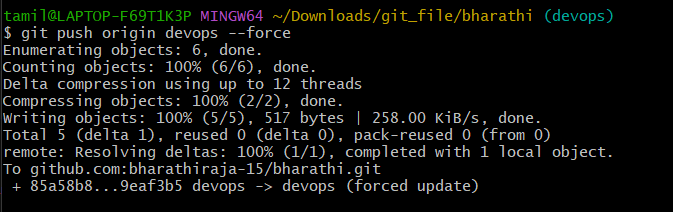


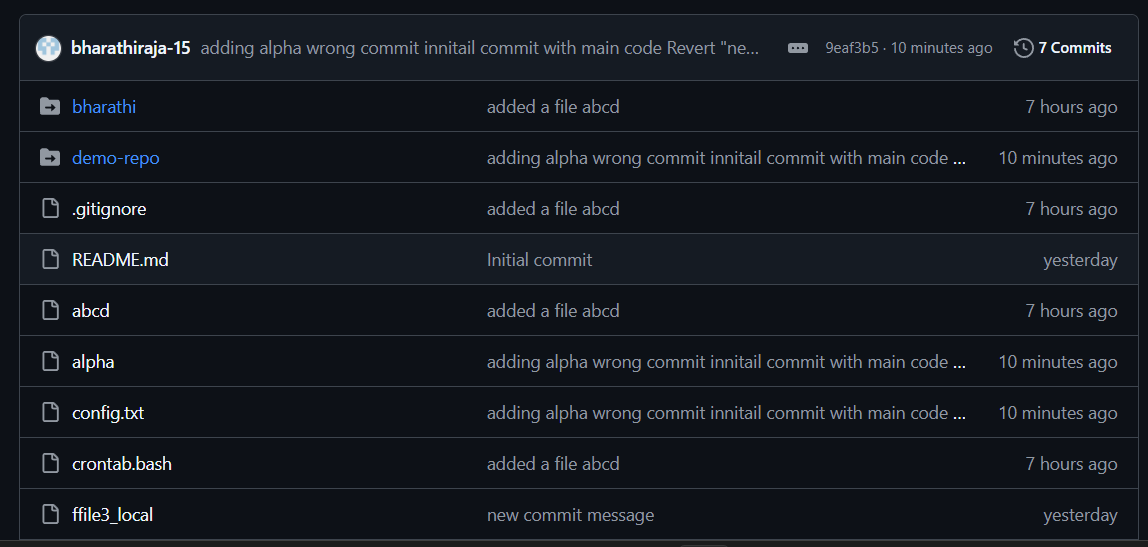
1. **Interactive Rebase**
   * Reorder and squash multiple commits into a single clean commit.





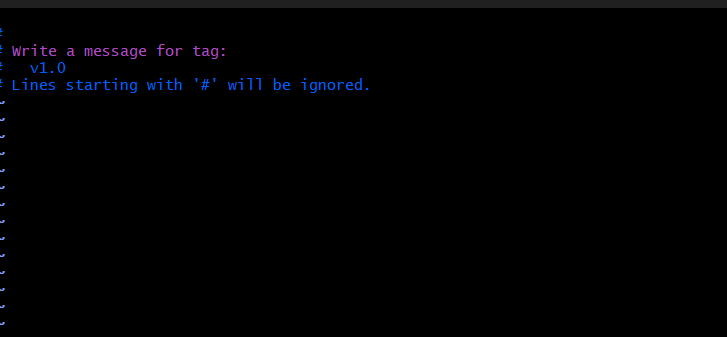


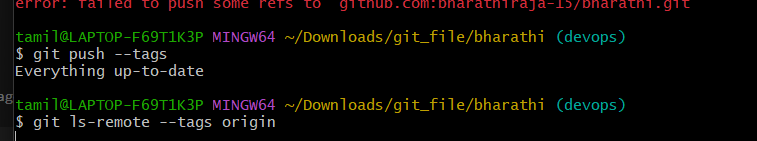




1. **Tagging & Release**
   * Create a version tag (v1.0), push it to GitHub, then delete and restore it.

git tag -a v1.0 -m "Release v1.0: initial stable release"



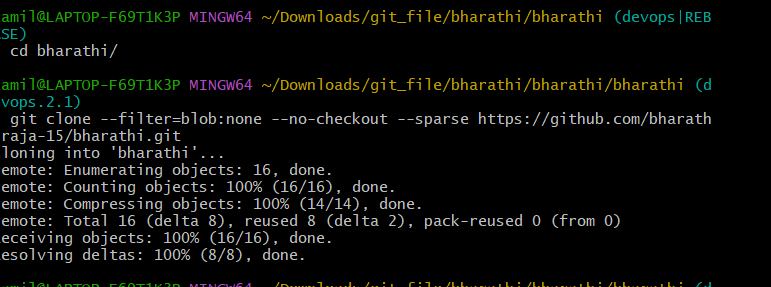


1. **Clone with Sparse Checkout**
   * Clone only a subdirectory of a repo using sparse checkout

# 2) Enter the repo

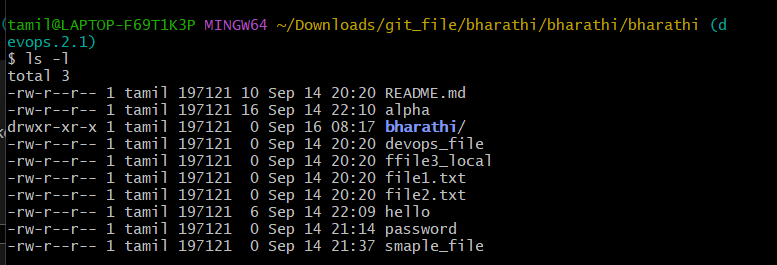
cd devops-hub-sparse

* + .



# 3) Initialize sparse-checkout in "cone" mode (recommended)

git sparse-checkout init --cone



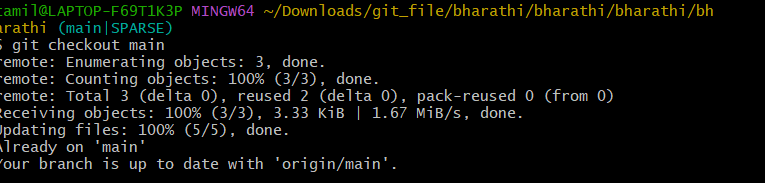
# 4) Select only the 'git' folder

git sparse-checkout set git



# 5) Checkout the branch you need (main in your case)

git checkout main



* Only the /git folder (and its files) will be pulled into your working directory.
* The rest of the repo stays hidden.
* You still have a .git folder, so you can commit/push changes if needed.

Check the tag/branch state:

git branch

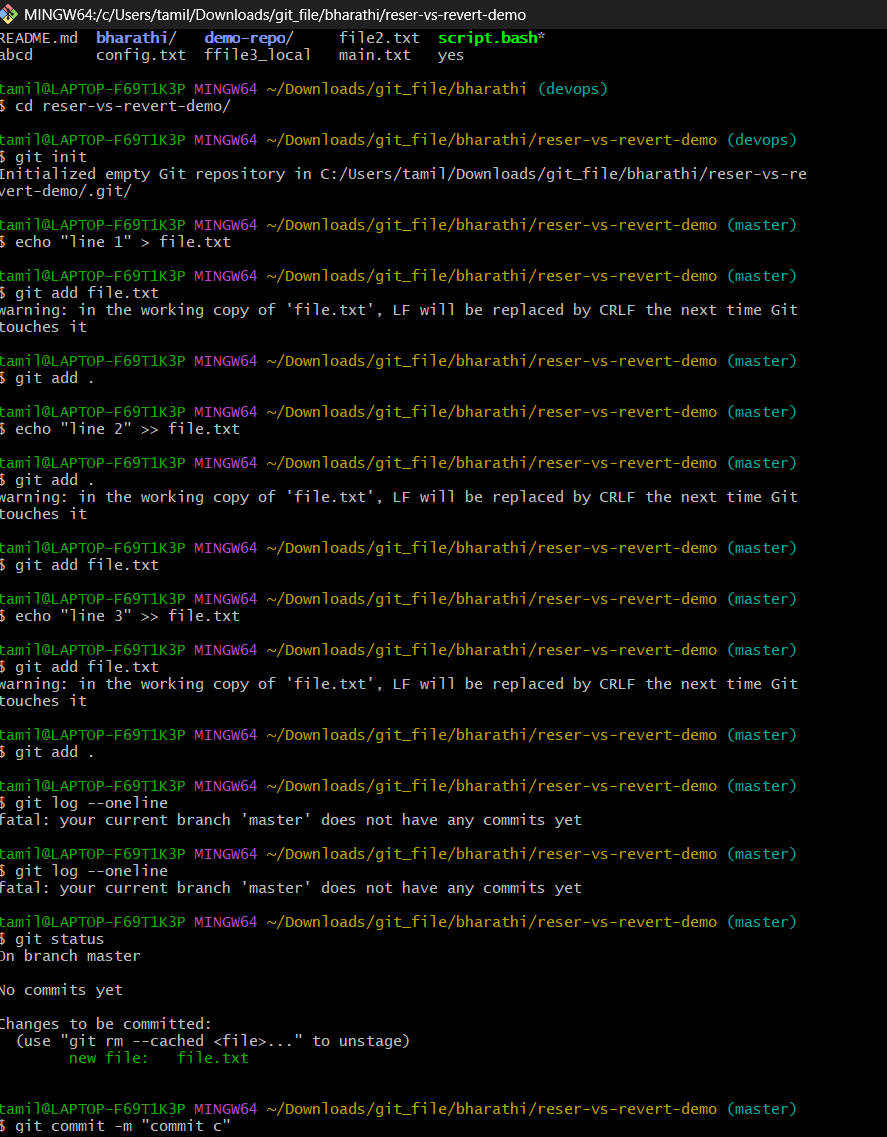
git log --oneline -- git/

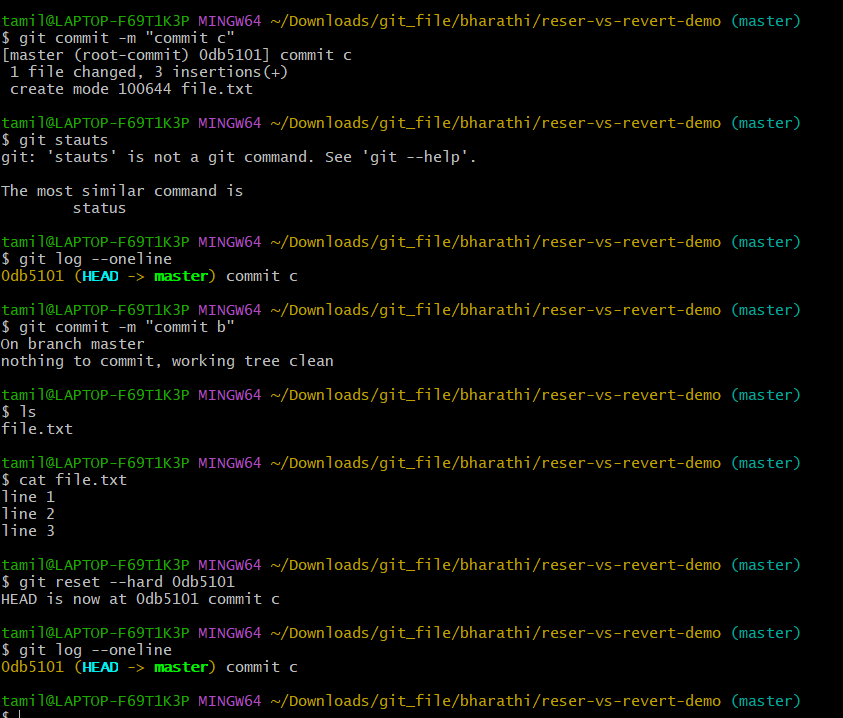
**⚡ Tips**

* If later you want another folder, e.g. /shell:
* git sparse-checkout set git shell
* To disable sparse-checkout and bring the full repo back:
* git sparse-checkout disable

git checkout main

1. **Reset vs Revert Challenge**
   * Demonstrate the difference between git reset --hard and git revert in a repo.

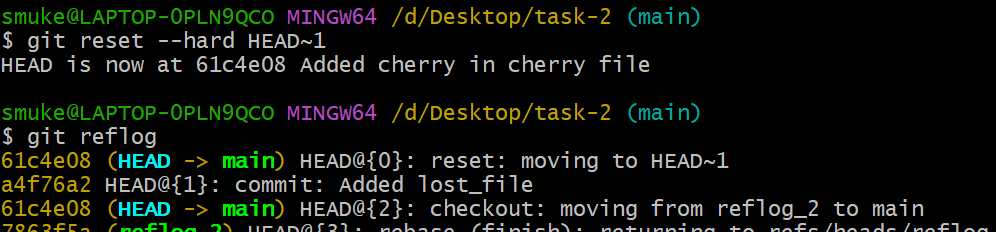




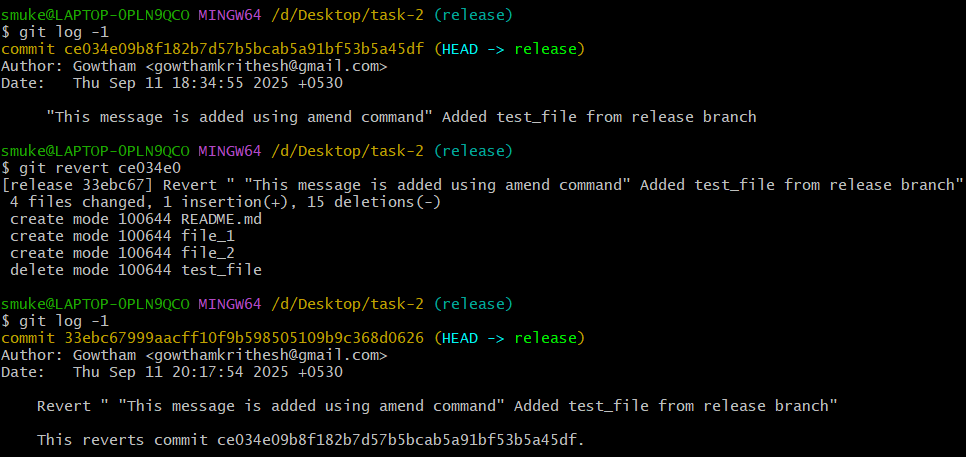
1. Reset vs Revert

**Reset**

**git reset --hard commit\_id** will rewrites the project history. It moves the HEAD pointer and the current branch reference to the specified **commit.**

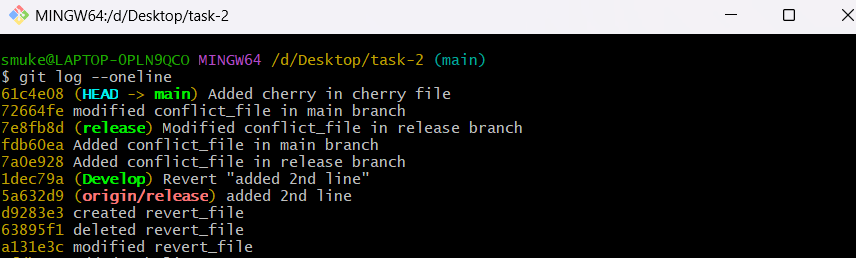


**Revert**

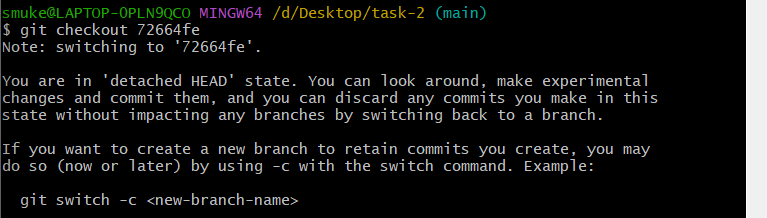
**git revert commit\_id** creates a new commit that undoes the changes introduced by the specified **commit.** 

1. Detached HEAD challenge

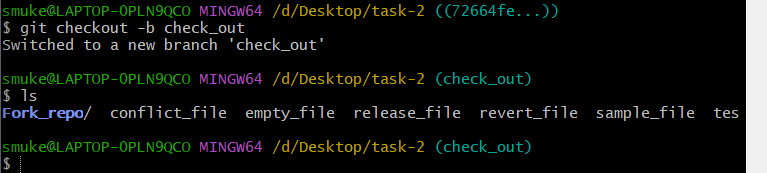
Step 1: Use **git log –online** to get the commit id.



Step 2: Use **git checkout commit\_id** to switch to that commit\_id.

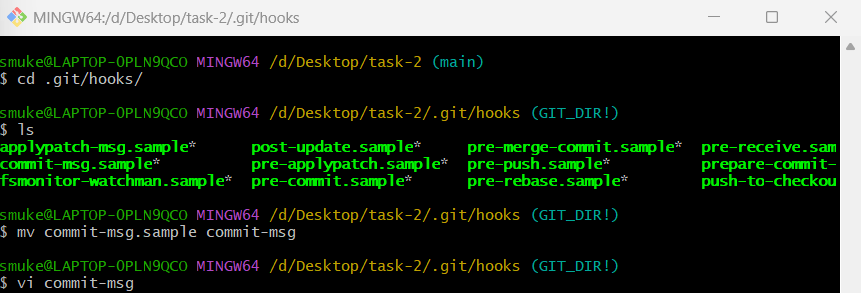


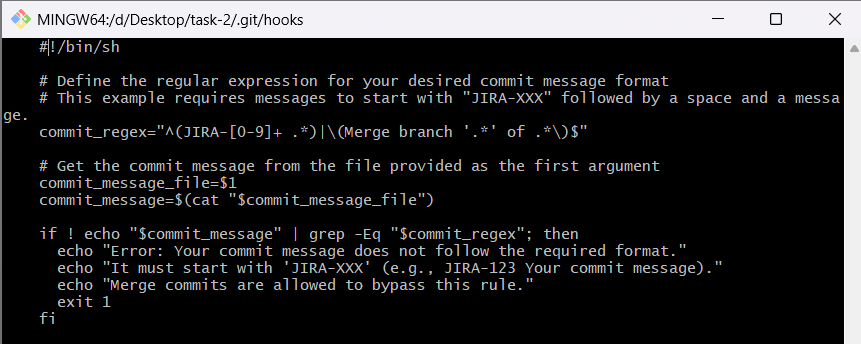
Step 3: Use **git checkout -b new\_branch\_name** to create new branch and retain commit.



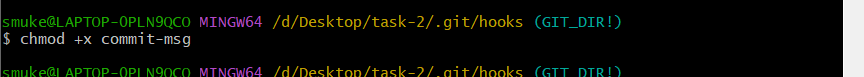
1. Git Hooks Challenge

Step 1: Use **cd .git/hooks/** and enter the scrip in **commit-msg** file

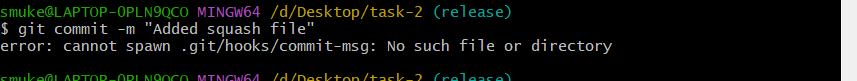




Step 2: Use **chmod +x commit-msg** to give executable permission to the file.



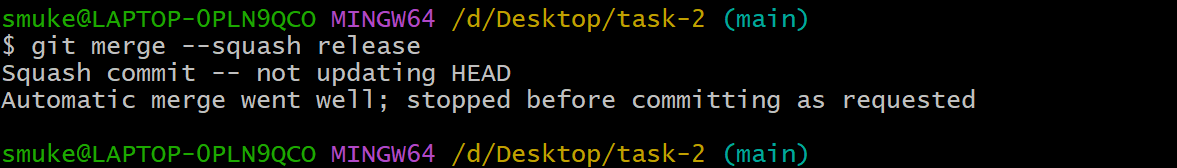
Now, whenever you attempt to commit, this commit-msg hook will automatically check if your commit message adheres to the specified "JIRA-XXX" format or not. If doesn’t commit will be rejected with error message.

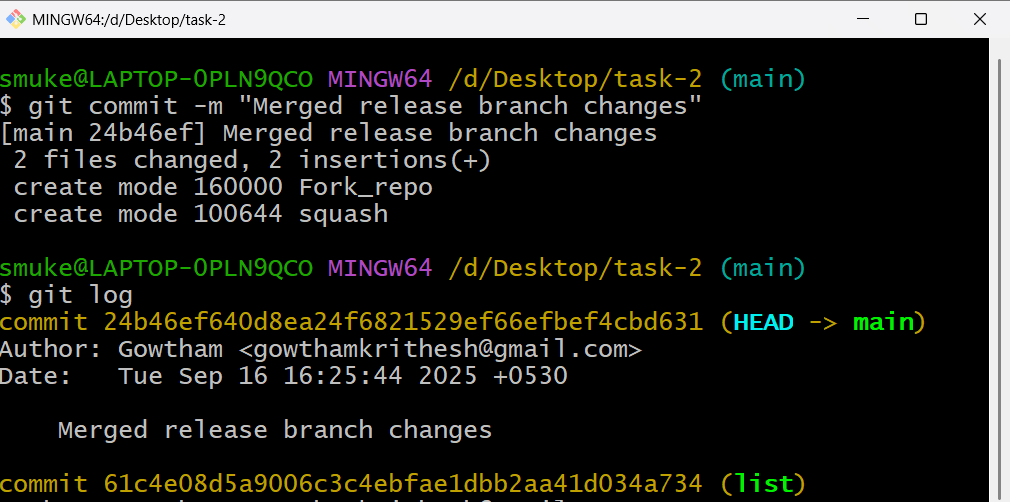


1. Squash Merge vs Rebase Merge.

**Squash Merge**

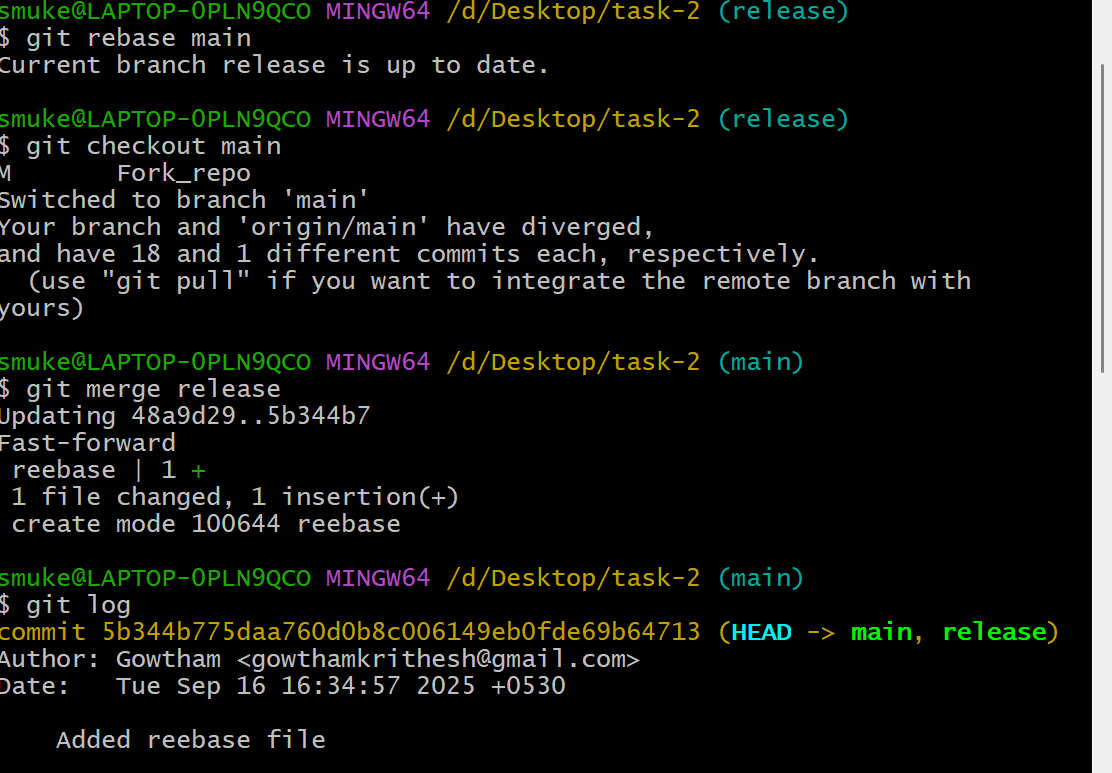
A squash merge will consolidate all commits from other branch into a single new commit on the target branch.





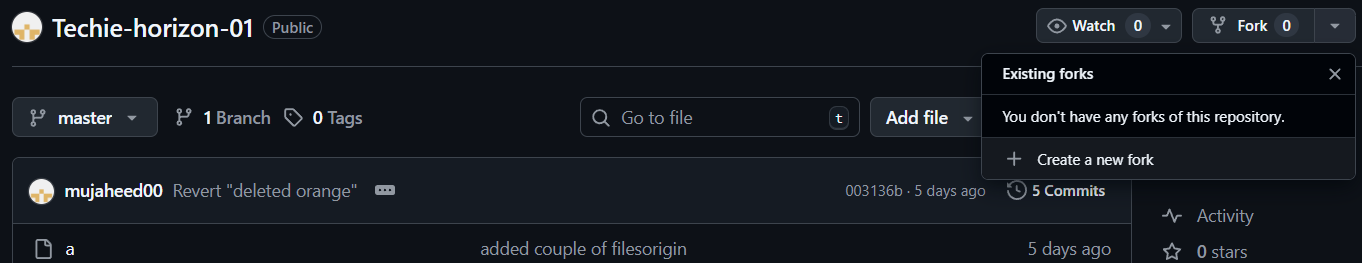
**Rebase Merge**

A rebase merge rewrites the commit history of the other branch by moving its base to the tip of the target branch.

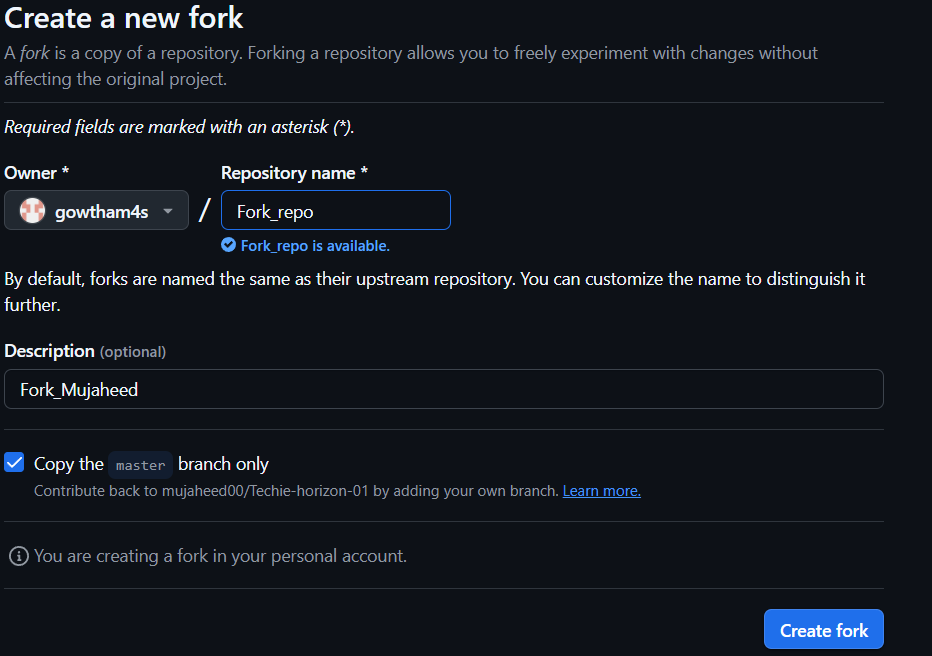


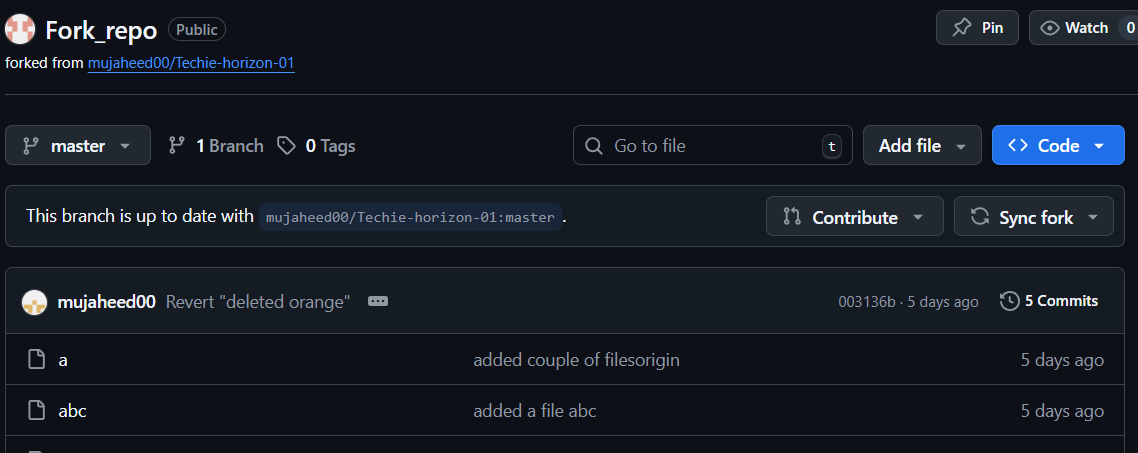
13. Fork & Pull request workflow

Step 1: open other repo click Fork 🡪create a new fork

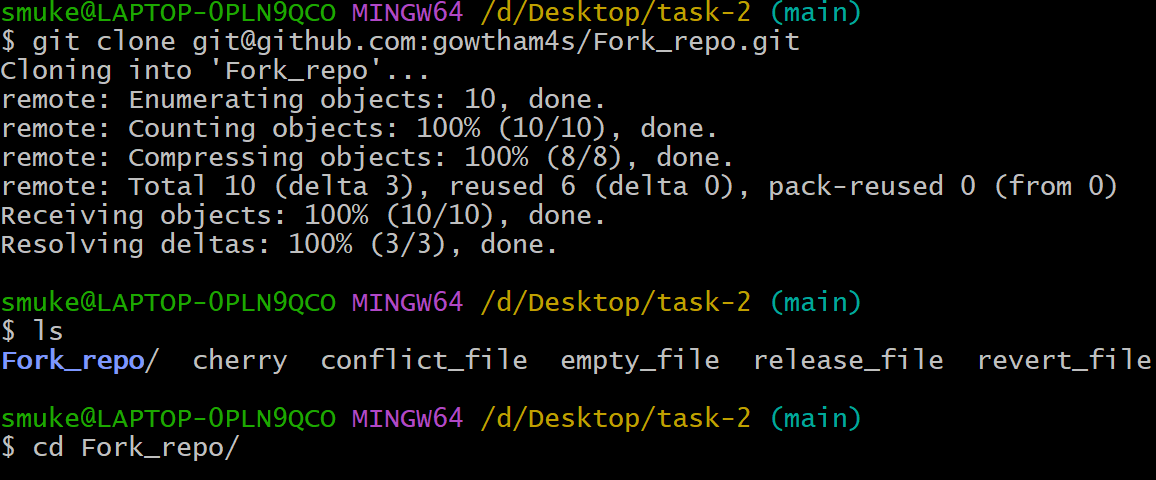


Step 2: Give repo name and click create fork





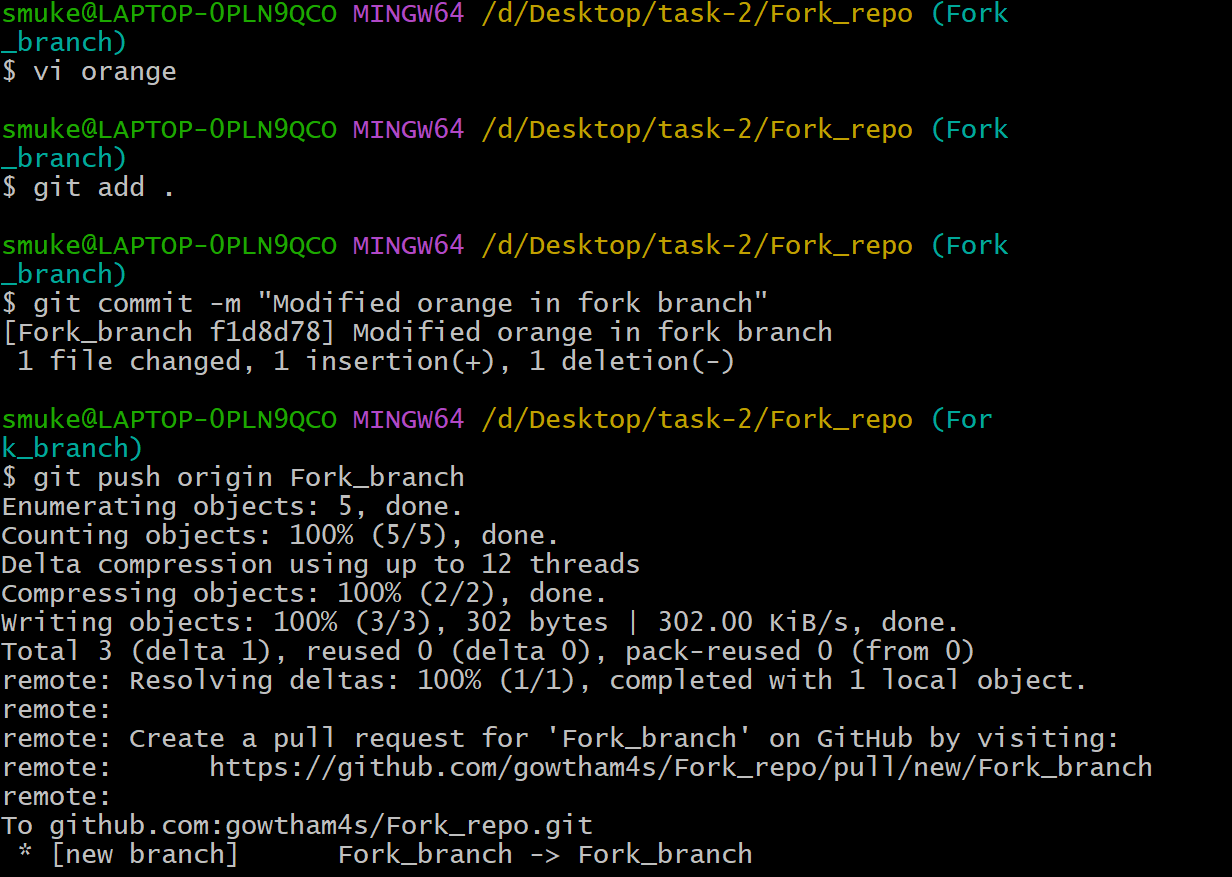
Step 3: Now in Git Enter **git clone fork\_repo\_url** and **cd Fork\_repo\_name**



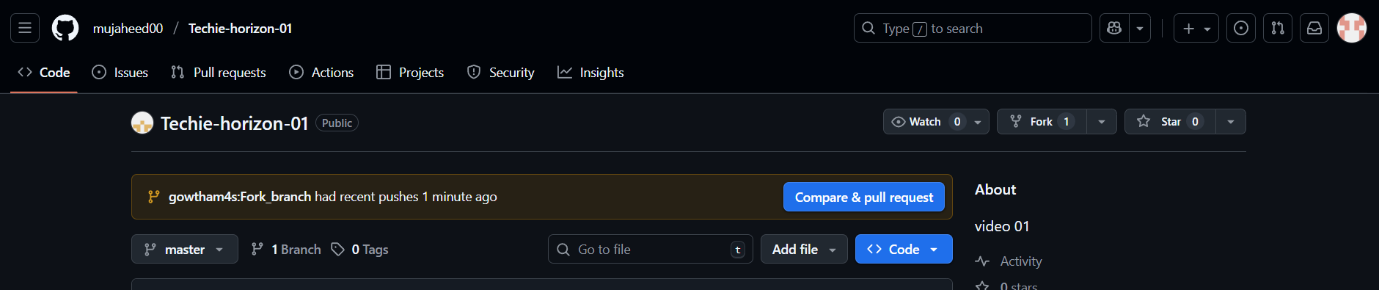
Step 4: Edit any file and push it.





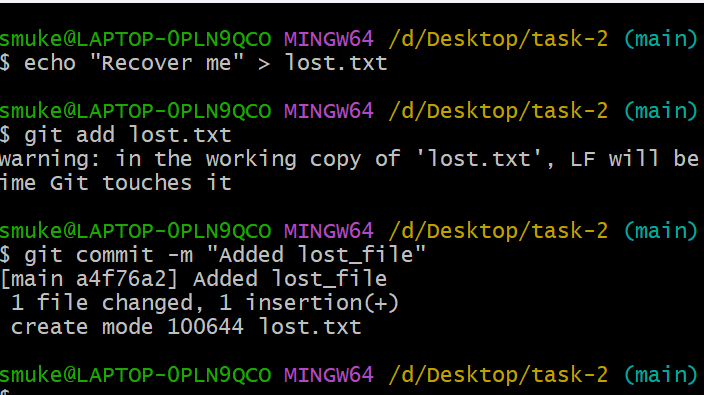


Step 5: Now go to original repo there you can see **compare & pull request**

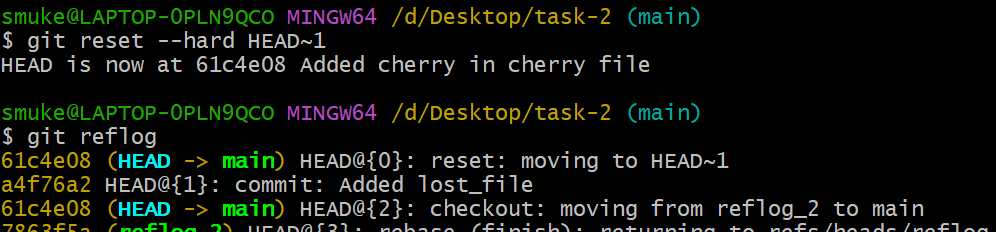
****

14. Recover lost commit.

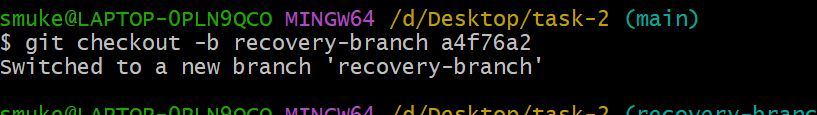
Step 1: Create a file and commit it.



Step 2: Use **git reset --hard HEAD~1** to revert one commit back and **git reflog** to list commit.



Step 3: Use **git checkout -b recovery-branch commit\_id** to create new branch at that commit



Step 4: Use **git reset --hard commit\_id** to move head back to that commit

