Git Interview most ask question.

1) This When we create Repo what are thing we need do for clone and commit and push. (Interview to understand he know s git basic will check)

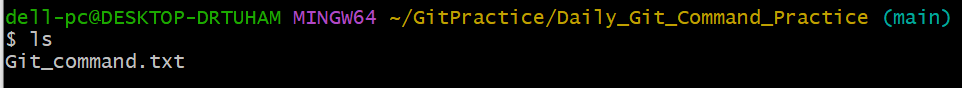
Describe the question..

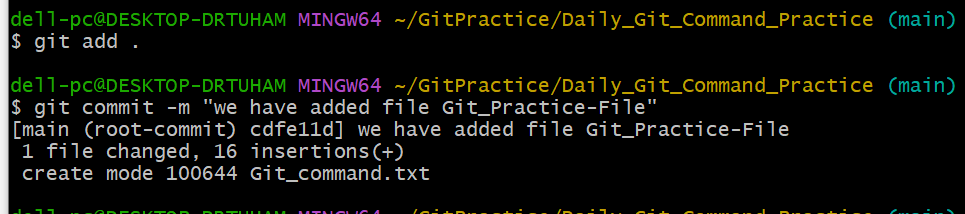
Ans - When we are clone clone repo form remote registry like Git

Fist we create folder workspace like any Project Name or something as per our needs

git Init - git init doesn’t modify or delete any of your existing files. It simply adds version control capabilities to the folder.

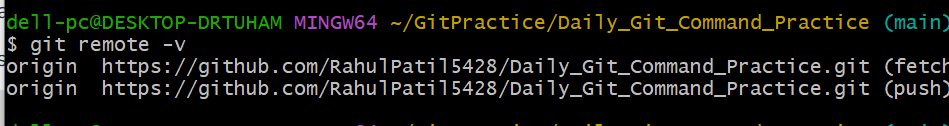
git add . When you want to stage all changes in the current directory and its subdirectories





Before Push code Repo check orgin repo URL on our workspace .

Command :- git origin -V



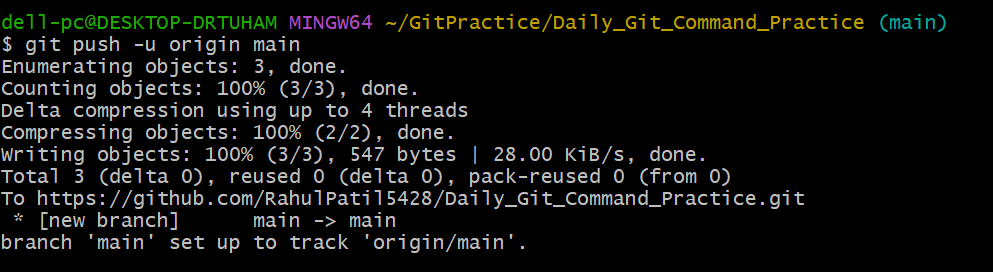
If any one not show the remote URL So we can add Below Command.

Command :- git remote add origin https://repourl.git

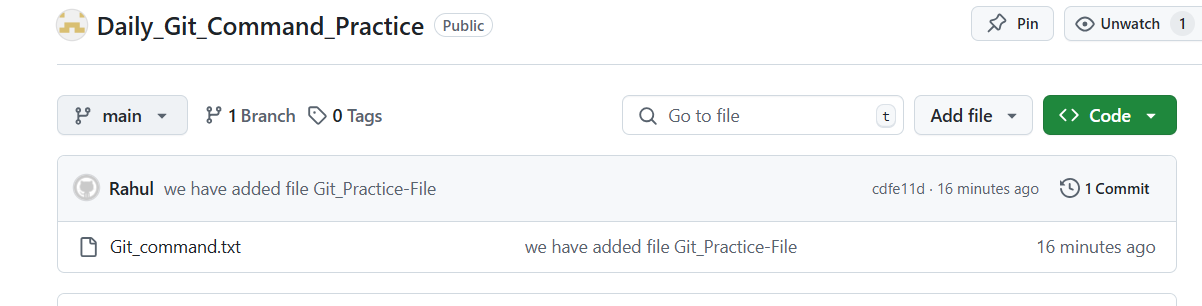
If our commit is done and the we can push code the our repo.

Command :- git push -u origin main ( you can define any branch name )

It will ask the Username and password or any access token .



Let’s Verify we create file GitHub account .

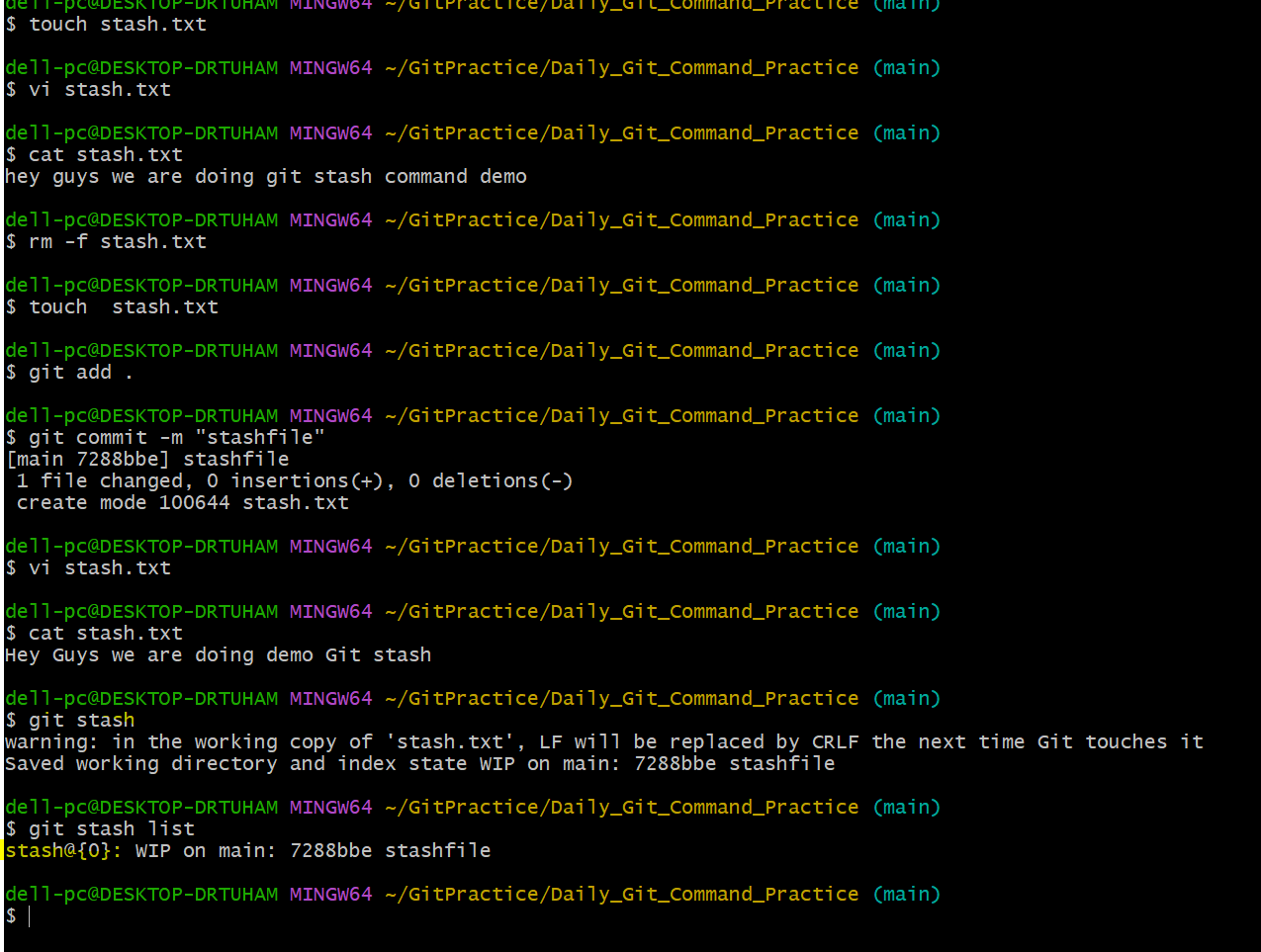


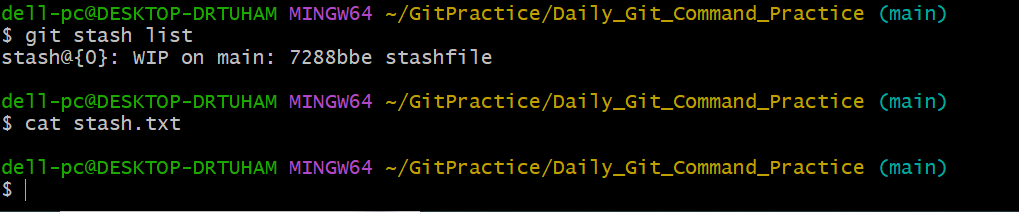
2) What is Git stash.

used to **temporarily save changes** in your working directory that aren’t ready to be committed. This allows you to **switch contexts** or work on something else (like a bug fix or feature) without losing the work you've done so far.

Example :- temporarily save changes our unfinished work we want save in stash our work will not delete or erase.

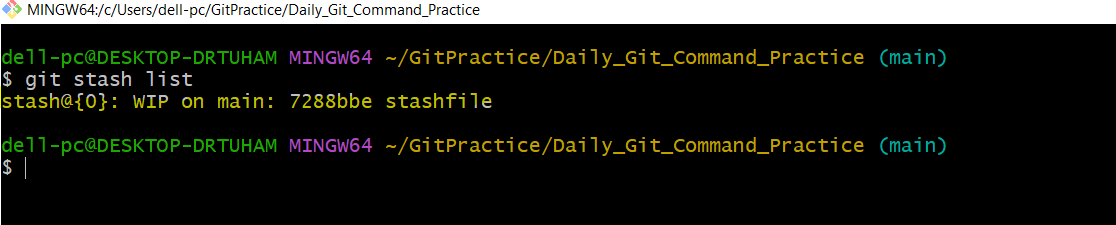
**We can create file for testing for perform stash command use.**

1. **We Created File blank file**
2. **Git add .**
3. **Git commit**
4. **We edit the file added some text on file**
5. **This is our unfinish work don’t loose the work**
6. **We are using command ( git stash )**
7. **Add in stash area file now**
8. 
9. **We can’t see work on file ( Lets verify )**



**Its blank its show nothing don’t worry about the work it will save stash area.**

**Command:- git stash list ( show the how many stash are available )**



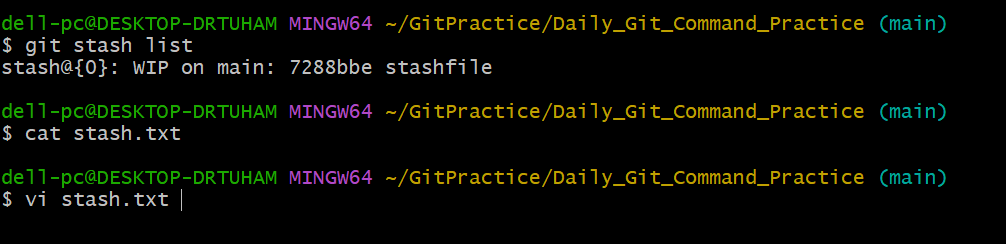
**stash@{0} -- Recently create stash**

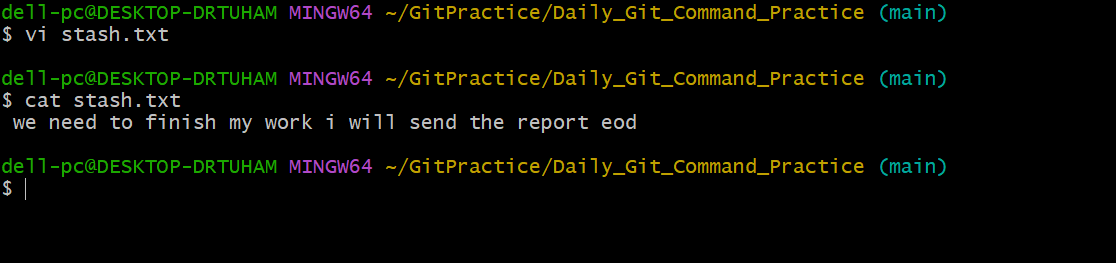
**main – is branch**

**7288bbe -- stash was made after the commit identified**

**Stashfile -- is file as we stash ( in your case any code file )**

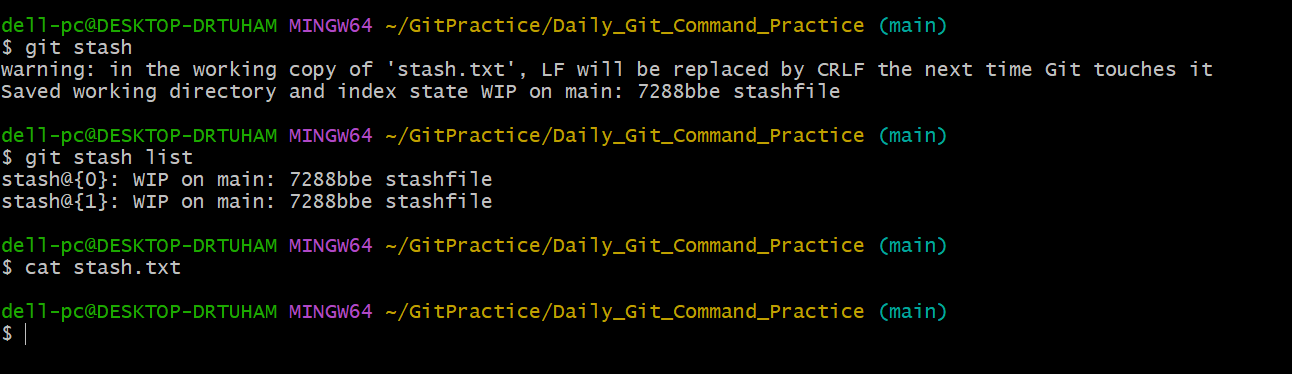
**Lets edit file again unfinishwork we can complete work**



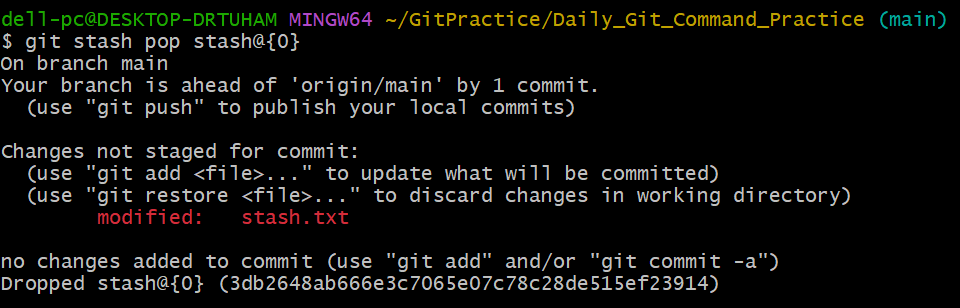


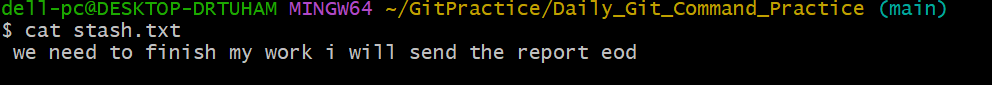
**We add some date on file**

**Add in stash**



**No text is visible**





**Command :- git stash apply ( it si only apply ) git stash pop ( its used for apply and remove stash form stash list.**

**For remove stash command**

**Git stash drop stash id (like number o, 1)**

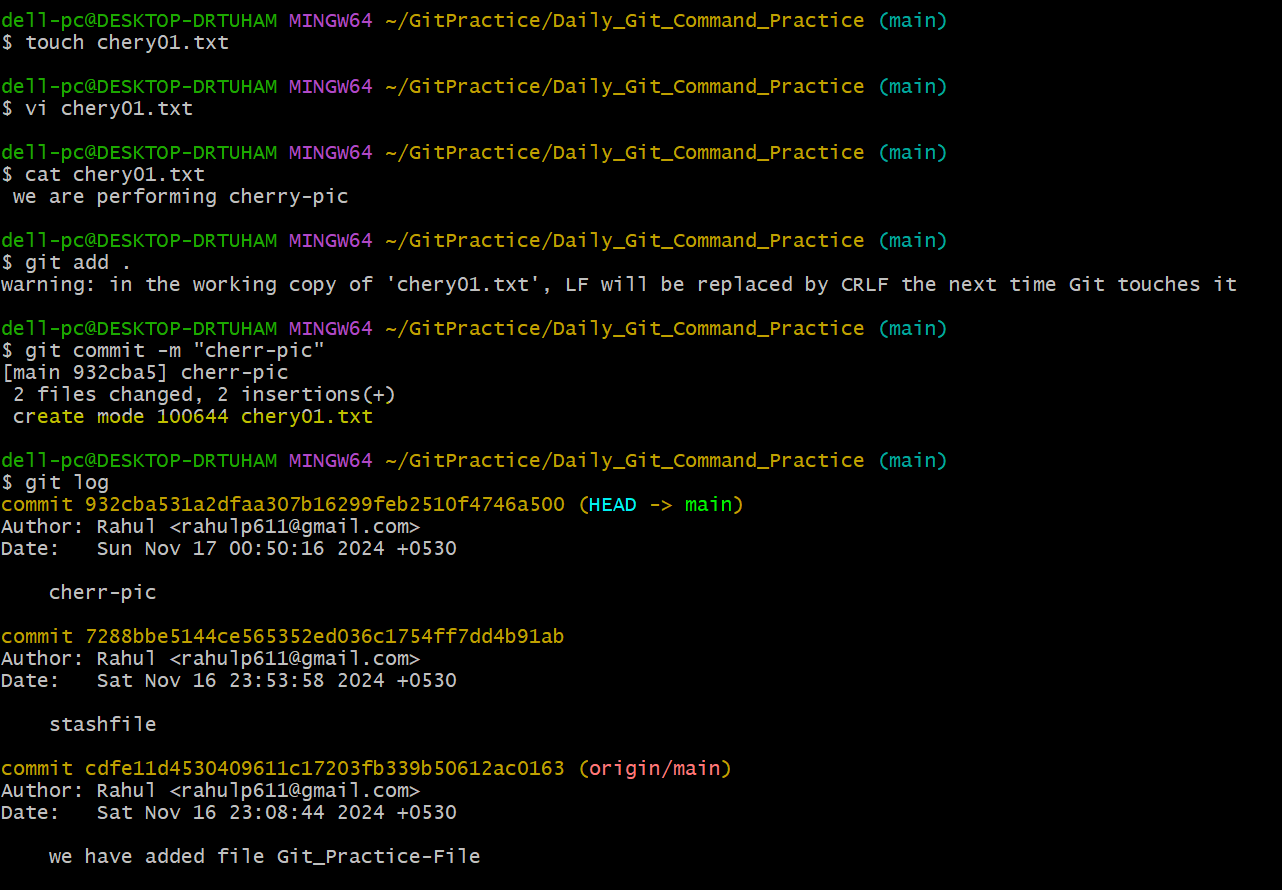
**Git stash clear ( this will all stash )**

**3) What is Chery-pick in Git and why used .**

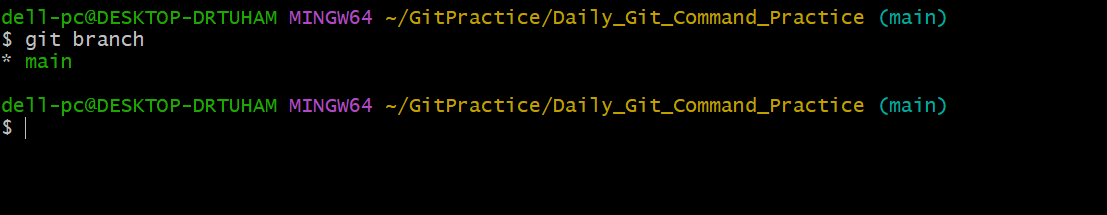
**git cherry-pick is a Git command used to apply a specific commit from one branch onto another branch. It allows you to "cherry-pick" (select) specific changes from a commit, instead of merging or rebasing an entire branch.**

**Example :- If a bug fix is committed on one branch (e.g., develop) and you need that fix in another branch (e.g., main) without merging all other changes, you can cherry-pick just the bug fix commit.**

**Now will created file for the test .**

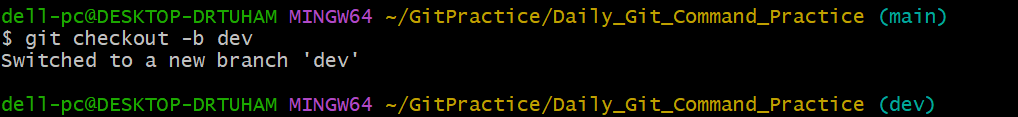


**We have applied all this command on main branch**

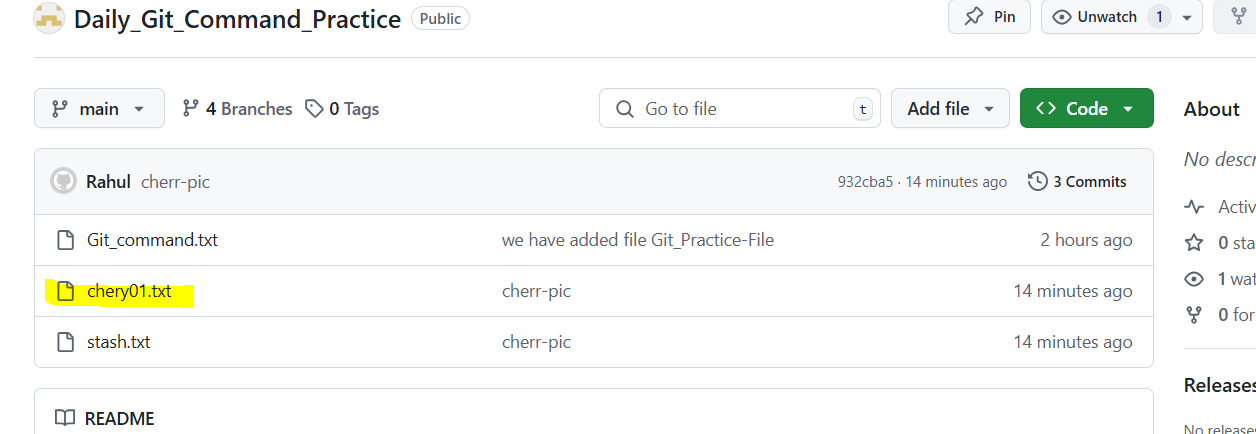


**We need to create branch for this commit should apply with new branch**

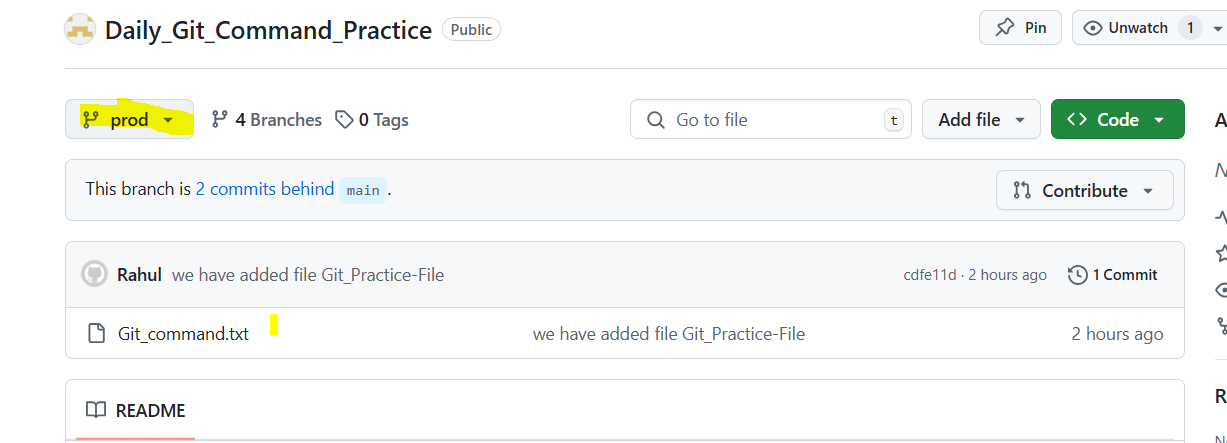
**Command :- git checkout -b dev**



**We have push code in main branch we see in console we have file called chery01.txt**



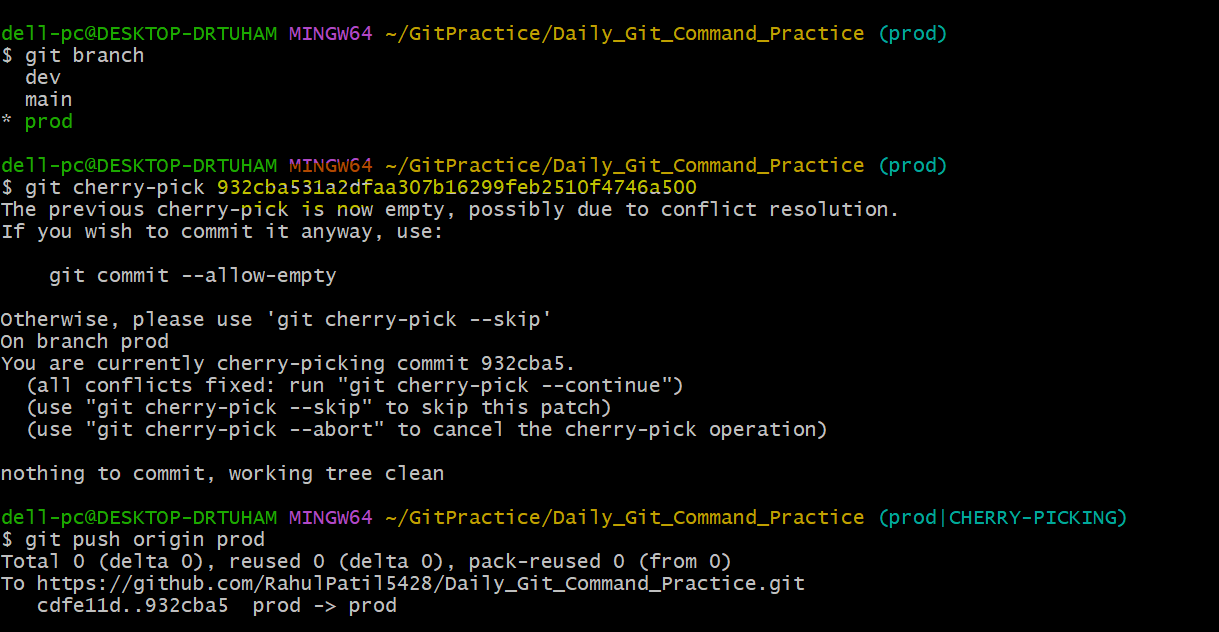
**We want Merge on prod branch specific commit we can see the console which are the file are availed**



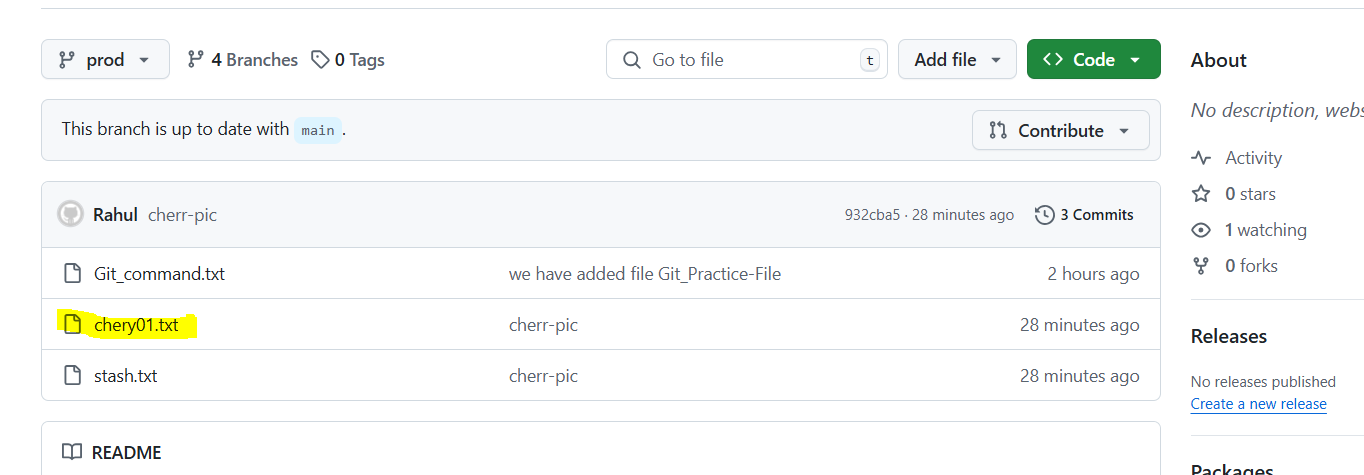
**There is no file are available cheryy01 file**

**Now Checkout branch in prod**

**Command: - git cherry-pick 932cba531a2dfaa307b16299feb2510f4746a500**



**we see in the console**



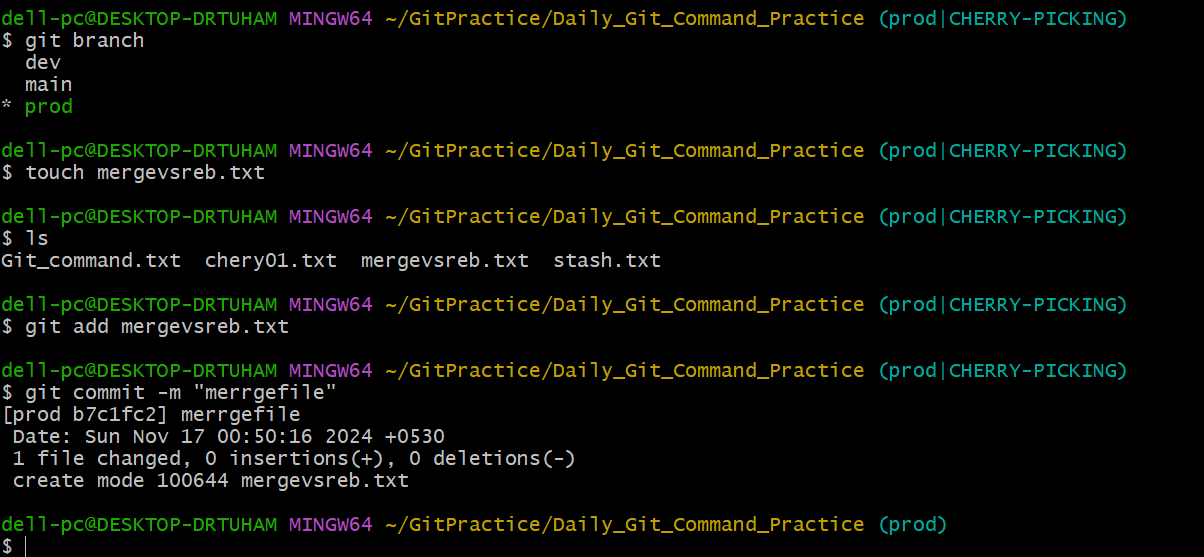
**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

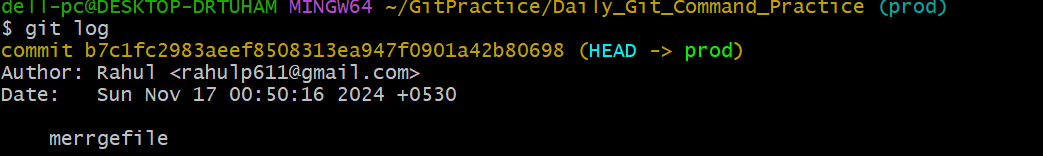
**2) What is Merge and Rebase.**

**Merge:- Merge combines the changes from one branch (e.g., feature) into another branch (e.g., main) by creating a new merge commit that ties together the histories of both branches**

**( while merge create Extra Commit ID )**

**Now I am in Prod Branch ( will create new file for test merge )**

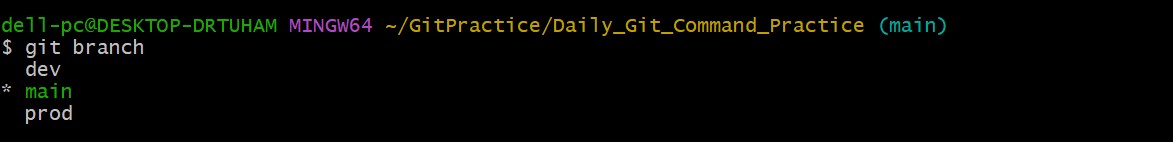


**We can see the log now :-** 

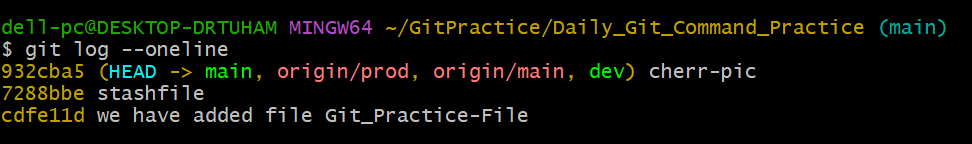
**Now go to main branch**

**Command :- git checkout main**

**Now I am at main branch**

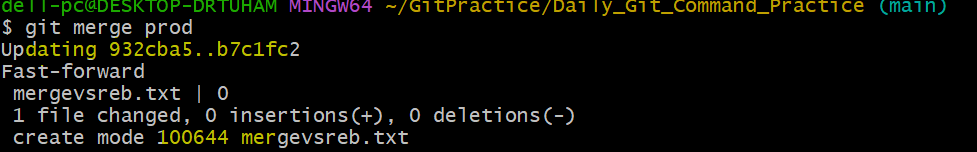


**Git log we can see other commit**

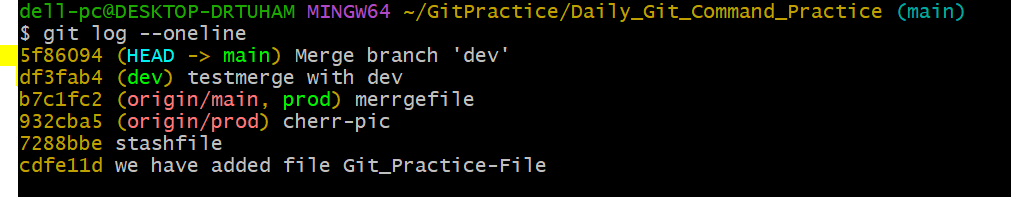


**Let start now merge.**

**Command :- git merge prod**



**We can see the log with 1 extra commit available.**



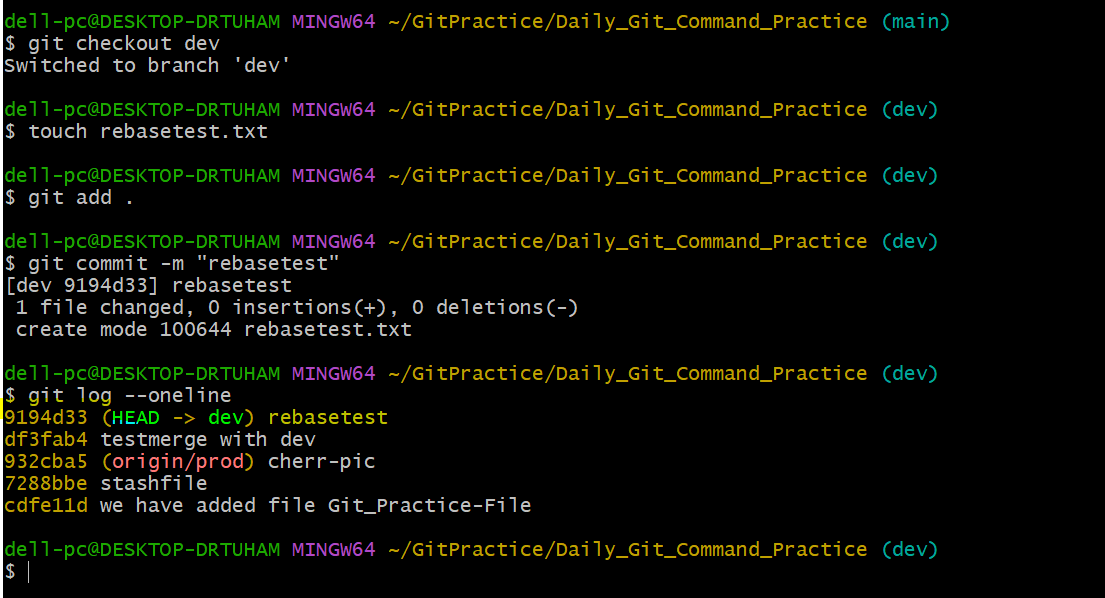
**Merge extra commit denote that merge has taken place**

**5f86094 (HEAD -> main) Merge branch 'dev'**

**What is Rebase?**

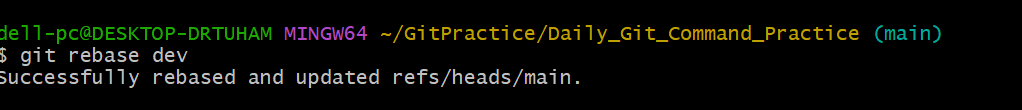
**Rebase integrates changes from one branch into another by replaying commits from one branch onto another, resulting in a linear commit history**

**Example: - Git takes the commits from your branch and replays them on top of the target branch (e.g., main), effectively creating a new, linear history.**

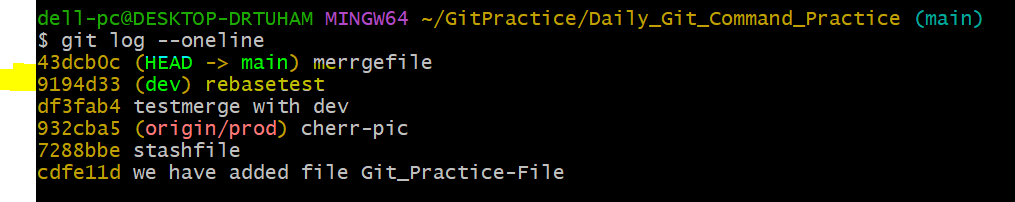


**Now Checkout main branch**

**Command: - git checkout main**



**See the log there is no extra commit.**



**Rebase like copy and paste.**