

- Git Challenges

1. Resolve Merge Conflicts:

a. Create a merge conflict intentionally (two users editing the same line).

```
MINGW64:/c/Users/DELL/Desktop/project-1/README.md

DELL@DESKTOP-RCNKJC4 MINGW64 ~/Desktop/project-1/README.md (main)
$ git pull origin release
From https://github.com/Mr-Devops777/README.md
 * branch      release    -> FETCH_HEAD
Auto-merging file2
CONFLICT (add/add): Merge conflict in file2
Automatic merge failed; fix conflicts and then commit the result.

DELL@DESKTOP-RCNKJC4 MINGW64 ~/Desktop/project-1/README.md (main|MERGING)
$ ls
README.md  file1  file2

DELL@DESKTOP-RCNKJC4 MINGW64 ~/Desktop/project-1/README.md (main|MERGING)
$ cat file2
<<<<<<< HEAD
hello techies
=====
hello wolrd
>>>>>>> 71f812b3923db5e182df1193b177766b3a3e45ad
```

or

```
MINGW64:/c/Users/DELL/Desktop/project-1/README.md

DELL@DESKTOP-RCNKJC4 MINGW64 ~/Desktop/project-1/README.md (main)
$ git pull origin release
From https://github.com/Mr-Devops777/README.md
 * branch      release    -> FETCH_HEAD
Auto-merging file2
CONFLICT (content): Merge conflict in file2
Automatic merge failed; fix conflicts and then commit the result.

DELL@DESKTOP-RCNKJC4 MINGW64 ~/Desktop/project-1/README.md (main|MERGING)
$ ls
README.md  file1  file2

DELL@DESKTOP-RCNKJC4 MINGW64 ~/Desktop/project-1/README.md (main|MERGING)
$ cat file2
hello techies
hello wolrd
<<<<<<< HEAD
hello everyone
=====
hello devops team
hello cloud engineers
>>>>>>> fca8d8dcc4822391c8faa84d1fa847d6564077b8
```

- I've added user name and email with (git config --global user.name user-1) (git config --global user.email user1@gmail.com)
- then i cloned a repo to my local machine there i created the file2 in main branch with some content in it and added to tracking area and committed and pushed to central
- then i've added one more user and email (git config --global user.name user-2)(git

config --global user.email user2@gmailcom)

- these user-2 also working on same repo in release branch where he created a file with same name as in main branch and added to tracking area then committed and pushed to central
- again user-1 swithc back and trying to pull the release branch in main the merge conflict arises in file2
- basically the merge will merge the data of two identical files (release-branch/file2 & main-branch/file2)
- merging message wil appear beside our branch name like (main|merging)

b. Resolve the conflict and push the changes.

```
DELL@DESKTOP-RCNKJC4 MINGW64 ~/Desktop/project-1/README.md (main|MERGING)
$ cat file2
<<<<<< HEAD
hello techies
=====
hello wolrd
>>>>>> 71f812b3923db5e182df1193b177766b3a3e45ad

DELL@DESKTOP-RCNKJC4 MINGW64 ~/Desktop/project-1/README.md (main|MERGING)
$ vi file2

DELL@DESKTOP-RCNKJC4 MINGW64 ~/Desktop/project-1/README.md (main|MERGING)
$ vi file2

DELL@DESKTOP-RCNKJC4 MINGW64 ~/Desktop/project-1/README.md (main|MERGING)
$ cat file2
hello techies
hello wolrd

DELL@DESKTOP-RCNKJC4 MINGW64 ~/Desktop/project-1/README.md (main|MERGING)
$ git status
On branch main
You have unmerged paths.
  (fix conflicts and run "git commit")
  (use "git merge --abort" to abort the merge)

Unmerged paths:
  (use "git add <file>..." to mark resolution)
    both added:   file2

no changes added to commit (use "git add" and/or "git commit -a")

DELL@DESKTOP-RCNKJC4 MINGW64 ~/Desktop/project-1/README.md (main|MERGING)
$ git add .

DELL@DESKTOP-RCNKJC4 MINGW64 ~/Desktop/project-1/README.md (main|MERGING)
$ git merge --continue
[main 3e8aec6] Merge branch 'release' of https://github.com/Mr-Devops777/README.md

DELL@DESKTOP-RCNKJC4 MINGW64 ~/Desktop/project-1/README.md (main)
$ cat file2
hello techies
hello wolrd
```

(or)

```

$ ls
README.md file1 file2

DELL@DESKTOP-RCNKJC4 MINGW64 ~/Desktop/project-1/README.md (main|MERGING)
$ cat file2
hello techies
hello wolrd
<<<<<<< HEAD
hello everyone
=====
hello devops team
hello cloud engineers
>>>>>>> fca8d8dcc4822391c8faa84d1fa847d6564077b8

DELL@DESKTOP-RCNKJC4 MINGW64 ~/Desktop/project-1/README.md (main|MERGING)
$ vi file2

DELL@DESKTOP-RCNKJC4 MINGW64 ~/Desktop/project-1/README.md (main|MERGING)
$ git merge --continue
error: Committing is not possible because you have unmerged files.
hint: Fix them up in the work tree, and then use 'git add/rm <file>'
hint: as appropriate to mark resolution and make a commit.
fatal: Exiting because of an unresolved conflict.
U      file2

DELL@DESKTOP-RCNKJC4 MINGW64 ~/Desktop/project-1/README.md (main|MERGING)
$ ls
README.md file1 file2

DELL@DESKTOP-RCNKJC4 MINGW64 ~/Desktop/project-1/README.md (main|MERGING)
$ git status
On branch main
You have unmerged paths.
  (fix conflicts and run "git commit")
  (use "git merge --abort" to abort the merge)

Unmerged paths:
  (use "git add <file>..." to mark resolution)
    both modified:   file2

no changes added to commit (use "git add" and/or "git commit -a")

DELL@DESKTOP-RCNKJC4 MINGW64 ~/Desktop/project-1/README.md (main|MERGING)
$ git add .

DELL@DESKTOP-RCNKJC4 MINGW64 ~/Desktop/project-1/README.md (main|MERGING)
$ git commit -m "fourth commit modified file2 after merging"
[main 782321d] fourth commit modified file2 after merging

```

- to resolve the conflict of merging two identical files we need to edit the file manually
- vi file name (modify the changes with proper team consent)
- git add file and git commit -m "file modified" then
- git merge --continue (to continue our merging process and finally the conflict resolved will appear)
- after these command an editor appears where simply we need to save or modify the merge message in it and save
- merge conflict resolved

2. Recover Deleted Branch

a. Delete a local branch and then recover it using the reflog.

```
MINGW64 ~/Desktop/project-1/README.md
DELL@DESKTOP-RCNKJC4 MINGW64 ~/Desktop/project-1/README.md (main)
$ git branch -a
* main
  release
  remotes/origin/HEAD -> origin/main
  remotes/origin/main

DELL@DESKTOP-RCNKJC4 MINGW64 ~/Desktop/project-1/README.md (main)
$ git checkout release
Switched to branch 'release'

DELL@DESKTOP-RCNKJC4 MINGW64 ~/Desktop/project-1/README.md (release)
$ ls
README.md file1 file2 file3

DELL@DESKTOP-RCNKJC4 MINGW64 ~/Desktop/project-1/README.md (release)
$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

DELL@DESKTOP-RCNKJC4 MINGW64 ~/Desktop/project-1/README.md (main)
$ git branch -d release
Deleted branch release (was 7c2afc6).

DELL@DESKTOP-RCNKJC4 MINGW64 ~/Desktop/project-1/README.md (main)
$ git branch -a
* main
  remotes/origin/HEAD -> origin/main
  remotes/origin/main

DELL@DESKTOP-RCNKJC4 MINGW64 ~/Desktop/project-1/README.md (main)
$ git reflog
fca8d8d (HEAD -> main, origin/main, origin/HEAD) HEAD@{0}: checkout: moving from release to main
7c2afc6 HEAD@{1}: checkout: moving from main to release
fca8d8d (HEAD -> main, origin/main, origin/HEAD) HEAD@{2}: checkout: moving from release to main
7c2afc6 HEAD@{3}: commit: created file3 in release
fca8d8d (HEAD -> main, origin/main, origin/HEAD) HEAD@{4}: checkout: moving from main to release
fca8d8d (HEAD -> main, origin/main, origin/HEAD) HEAD@{5}: reset: moving to fca8d8d
8146cf2 HEAD@{6}: revert: Revert "third commit modified file2"
fca8d8d (HEAD -> main, origin/main, origin/HEAD) HEAD@{7}: checkout: moving from release to main
fca8d8d (HEAD -> main, origin/main, origin/HEAD) HEAD@{8}: checkout: moving from main to release
fca8d8d (HEAD -> main, origin/main, origin/HEAD) HEAD@{9}: clone: from https://github.com/Mr-Devops777/README.md.git

DELL@DESKTOP-RCNKJC4 MINGW64 ~/Desktop/project-1/README.md (main)
$ git checkout -b release 7c2afc6
Switched to a new branch 'release'

DELL@DESKTOP-RCNKJC4 MINGW64 ~/Desktop/project-1/README.md (release)
$ ls
README.md file1 file2 file3
```

-
- create a branch using
git branch release (for creating a branch)
- git checkout release (for switching the branch)
- create a file and add it and commit it in release branch
- git checkout main (to switch the branch again to main)
- git branch -d release (to delete the branch) or git branch -D release (for forcibly deleting the branches)
- git branch -a (to check list of branches)
- git reflog (to get the commit id)
- git checkout -b release commit_id (to create a branch with all files)
- git branch -a (again check the list of branches)

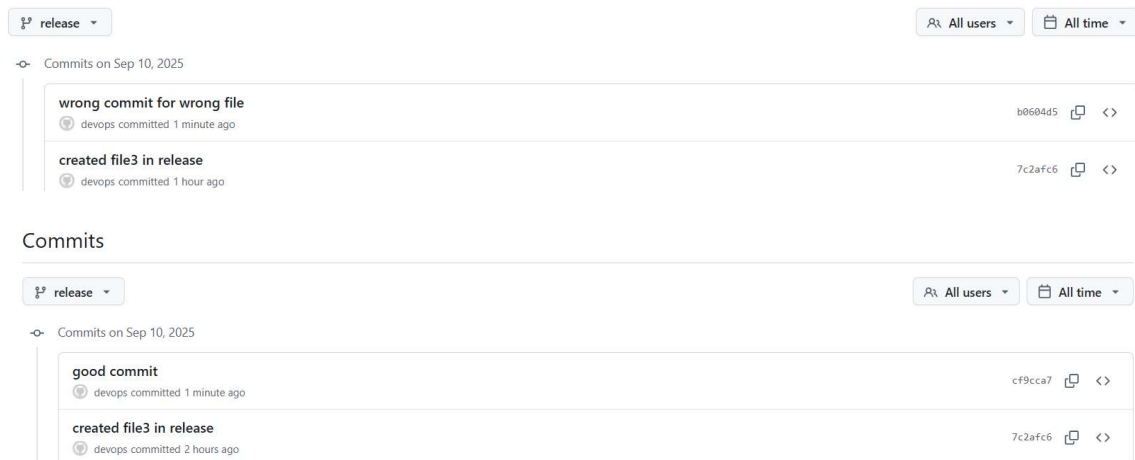
3. Undo Wrong Push

a. Push a wrong commit to GitHub, then undo it without losing history.

```
DELL@DESKTOP-RCNKJC4 MINGW64 ~/Desktop/project-1/README.md (release)
$ git log --oneline
b0604d5 (HEAD -> release, origin/release) wrong commit for wrong file
7c2afc6 created file3 in release
fca8d8d (origin/main, origin/HEAD, main) third commit modified file2
78b5ca8 second commit modified file2
71f812b first commit created file2 in release
5fac728 second commit modified file1 in release
d751c1f first commit created file1
603e667 Initial commit

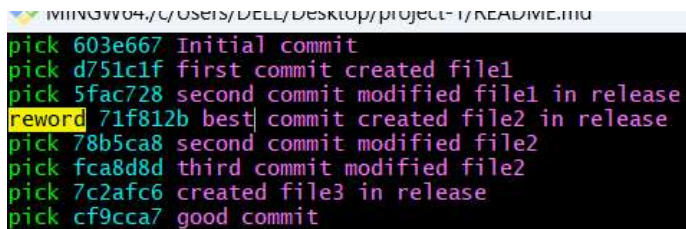
DELL@DESKTOP-RCNKJC4 MINGW64 ~/Desktop/project-1/README.md (release)
$ git commit --amend -m "good commit"
[release cf9cca7] good commit
Date: Wed Sep 10 19:42:35 2025 +0530
1 file changed, 1 insertion(+)
create mode 100644 wrongfile

DELL@DESKTOP-RCNKJC4 MINGW64 ~/Desktop/project-1/README.md (release)
$ git log --oneline
cf9cca7 (HEAD -> release) good commit
7c2afc6 created file3 in release
fca8d8d (origin/main, origin/HEAD, main) third commit modified file2
78b5ca8 second commit modified file2
71f812b first commit created file2 in release
5fac728 second commit modified file1 in release
d751c1f first commit created file1
603e667 Initial commit
```



- **git log --oneline** (check logs in both local and central)
- **git commit --amend -m "undo commit"** (it will directly replace the latest commit only it is not suitable for other commits which are already created)

- `git log --oneline` (again check the logs)
- `git push origin release --all` (to push all changes we're using all to make the central repo same as our local repo otherwise it will show some errors it will ask to fetch or pull)
- now you can check the commits in github
- **or**
- if i want to change any previous specific commit i'll use
- `git rebase -i --root`
- an editor file will appear there
- reword commit_id (in place of pick type reword and save)
- again one editor will open there change the commit and save :wq!
- `git log --oneline`



```

MINGW64: C:/Users/DELL/Desktop/project-1/README.MD
pick 603e667 Initial commit
pick d751c1f first commit created file1
pick 5fac728 second commit modified file1 in release
reword 71f812b best commit created file2 in release
pick 78b5ca8 second commit modified file2
pick fca8d8d third commit modified file2
pick 7c2afc6 created file3 in release
pick cf9cca7 good commit

```


MINGW64:/c/Users/DELL/Desktop/project-1/README.md

```
DELL@DESKTOP-RCNKJC4 MINGW64 ~/Desktop/project-1/README.md (release)
$ git log --oneline
cf9cca7 (HEAD -> release, origin/release) good commit
7c2afc6 created file3 in release
fca8d8d (origin/main, origin/HEAD, main) third commit modified file2
78b5ca8 second commit modified file2
71f812b first commit created file2 in release
5fac728 second commit modified file1 in release
d751c1f first commit created file1
603e667 Initial commit

DELL@DESKTOP-RCNKJC4 MINGW64 ~/Desktop/project-1/README.md (release)
$ git rebase -i --root
[detached HEAD 911e305] best commit created file2 in release
Date: Wed Sep 10 15:08:19 2025 +0530
1 file changed, 1 insertion(+)
create mode 100644 file2
Successfully rebased and updated refs/heads/release.

DELL@DESKTOP-RCNKJC4 MINGW64 ~/Desktop/project-1/README.md (release)
$ git log --oneline
8c0e299 (HEAD -> release) good commit
7e85829 created file3 in release
5fa16cf third commit modified file2
3010a49 second commit modified file2
911e305 best commit created file2 in release
5fac728 second commit modified file1 in release
d751c1f first commit created file1
603e667 Initial commit
```

4. AMEND A COMMIT

a. Make a commit, then add a missing file to it using git commit --amend.

MINGW64:/c/Users/DELL/Desktop/project-1/README.md

```
DELL@DESKTOP-RCNKJC4 MINGW64 ~/Desktop/project-1/README.md (release)
$ ls
README.md amendfile1 amendfile2 file1 file2 file3 wrongfile

DELL@DESKTOP-RCNKJC4 MINGW64 ~/Desktop/project-1/README.md (release)
$ git add .
warning: in the working copy of 'amendfile2', LF will be replaced by CRLF the next time Git touches it

DELL@DESKTOP-RCNKJC4 MINGW64 ~/Desktop/project-1/README.md (release)
$ git commit --amend -m "first commit added amendfile1 and amendfile2"
[release d99b199] first commit added amendfile1 and amendfile2
Date: Thu Sep 11 14:55:35 2025 +0530
2 files changed, 2 insertions(+)
create mode 100644 amendfile1
create mode 100644 amendfile2

DELL@DESKTOP-RCNKJC4 MINGW64 ~/Desktop/project-1/README.md (release)
$ git log --oneline
d99b199 (HEAD -> release) first commit added amendfile1 and amendfile2
8c0e299 good commit
7e85829 created file3 in release
5fa16cf third commit modified file2
3010a49 second commit modified file2
911e305 best commit created file2 in release
5fac728 second commit modified file1 in release
d751c1f first commit created file1
603e667 Initial commit
```

- create a file
- echo "creating first file for amend"amendfile1

- `git add amendfile1`
- `git commit -m "first commit added amendfile1"`
- then create one more file
- `echo "creating second file for amend" >amendfile2`
- `git add amendfile2`
- `git log --oneline` (now check the commits where you want to make a single commit for two files)
- `git commit --amend -m "first commit added amendfile1 and amendfile2"`
- `git log --oneline` (now you can check two files added in single commit)

5. Cherry-pick a Commit

a. 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 8c0e299` (commit _id)
- `git log --oneline` (hence you can get the specific id in other branch)
-

6. Interactive Rebase

a. Reorder and squash multiple commits into a single clean commit.


```

DELL@DESKTOP-RCNKJ3C4 MINGW64 ~/Desktop/project-1/README.md (release)
$ git log --oneline
d99b199 (HEAD -> release) first commit added amendfile1 and amendfile2
8c0e299 good commit
7e85829 created file3 in release
5fa16cf third commit modified file2
3010a49 second commit modified file2
911e305 best commit created file2 in release
5fac728 second commit modified file1 in release
d751c1f first commit created file1
603e667 Initial commit

DELL@DESKTOP-RCNKJ3C4 MINGW64 ~/Desktop/project-1/README.md (release)
$ git rebase -i --root
[detached HEAD e76702b] first squash commit created file1
Date: Wed Sep 10 15:00:08 2025 +0530
1 file changed, 1 insertion(+)
create mode 100644 file1
[detached HEAD 327f01e] first squash commit created file1
Date: Wed Sep 10 15:00:08 2025 +0530
6 files changed, 7 insertions(+)
create mode 100644 amendfile1
create mode 100644 amendfile2
create mode 100644 file1
create mode 100644 file2
create mode 100644 file3
create mode 100644 wrongfile
Successfully rebased and updated refs/heads/release.

DELL@DESKTOP-RCNKJ3C4 MINGW64 ~/Desktop/project-1/README.md (release)
$ git log --oneline
327f01e (HEAD -> release) first squash commit created file1
603e667 Initial 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

7. Tagging & Release

a. Create a version tag (v1.0), push it to GitHub, then delete and restore it.

8. Clone with Sparse Checkout

a. Clone only a subdirectory of a repo using sparse checkout.

9. Reset vs Revert Challenge

a. Demonstrate the difference between git reset --hard and git revert in a repo.

```
MINGW64 ~/C:/Users/DELL/Desktop/project-1/README.md
DELL@DESKTOP-RCNKJC4 MINGW64 ~/Desktop/project-1/README.md (release)
$ git log --oneline
c992271 (HEAD -> release, origin/release) added revert.txt
327f01e first squash commit created file1
603e667 Initial commit

DELL@DESKTOP-RCNKJC4 MINGW64 ~/Desktop/project-1/README.md (release)
$ ls
README.md  amendfile1  amendfile2  file1  file2  file3  revert.txt  wrongfile

DELL@DESKTOP-RCNKJC4 MINGW64 ~/Desktop/project-1/README.md (release)
$ rm revert.txt

DELL@DESKTOP-RCNKJC4 MINGW64 ~/Desktop/project-1/README.md (release)
$ git status
On branch release
Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        deleted:    revert.txt

no changes added to commit (use "git add" and/or "git commit -a")

DELL@DESKTOP-RCNKJC4 MINGW64 ~/Desktop/project-1/README.md (release)
$ git add revert.txt

DELL@DESKTOP-RCNKJC4 MINGW64 ~/Desktop/project-1/README.md (release)
$ ls
README.md  amendfile1  amendfile2  file1  file2  file3  wrongfile

DELL@DESKTOP-RCNKJC4 MINGW64 ~/Desktop/project-1/README.md (release)
$ git commit -m "deleted revert.txt"
[release 7f89b62] deleted revert.txt
1 file changed, 1 deletion(-)
delete mode 100644 revert.txt

DELL@DESKTOP-RCNKJC4 MINGW64 ~/Desktop/project-1/README.md (release)
$ git log --oneline
7f89b62 (HEAD -> release) deleted revert.txt
c992271 (origin/release) added revert.txt
327f01e first squash commit created file1
603e667 Initial commit

DELL@DESKTOP-RCNKJC4 MINGW64 ~/Desktop/project-1/README.md (release)
$ git revert 7f89b62
[release 1784c8c] Revert "added again revert.txt"
1 file changed, 1 insertion(+)
create mode 100644 revert.txt

DELL@DESKTOP-RCNKJC4 MINGW64 ~/Desktop/project-1/README.md (release)
$ ls
README.md  amendfile1  amendfile2  file1  file2  file3  revert.txt  wrongfile

DELL@DESKTOP-RCNKJC4 MINGW64 ~/Desktop/project-1/README.md (release)
$ git log --oneline
1784c8c (HEAD -> release) Revert "added again revert.txt"
7f89b62 deleted revert.txt
c992271 (origin/release) added revert.txt
327f01e first squash commit created file1
603e667 Initial commit
```

git revert

- echo "file for revert">revert.txt

- `git add revert.txt`
- `git commit -m "added revert.txt"`
- `rm revert.txt`
- `git add revert.txt`
- `git commit -m "deleted revert.txt"`
- `ls` (check the files)
- `git log --oneline`
- `git revert commit_id` (commit id of revert file added)
- vi editor will open there modify the revert message and save
- `ls` (now you can find the file which got deleted)
- `git log --oneline` (new commit id will be created with revert extension)

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 changes 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 difference of git reset and git revert is*
git reset : it will reset the changes till a specific commit by removing the history its not

safe to use)

- *git revert : it will modify the commit by keeping the history and creating one more commit for it)*

10. Detached HEAD Challenge

- a. Checkout a specific commit (detached HEAD state) and create a new branch from it.

```

DELL@DESKTOP-RCNKJC4 MINGW64 ~/Desktop/project-1/README.md (release)
$ git log --oneline
1784c8c (HEAD -> release) Revert "added again revert.txt"
7f89b62 deleted revert.txt
c992271 (origin/release) added revert.txt
327f01e first squash commit created file1
603e667 Initial commit

DELL@DESKTOP-RCNKJC4 MINGW64 ~/Desktop/project-1/README.md (release)
$ git checkout 603e667
Note: switching to '603e667'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by switching back to a branch.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -c with the switch command. Example:

    git switch -c <new-branch-name>

Or undo this operation with:

    git switch -

Turn off this advice by setting config variable advice.detachedHead to false

HEAD is now at 603e667 Initial commit

DELL@DESKTOP-RCNKJC4 MINGW64 ~/Desktop/project-1/README.md ((603e667...))
$ git checkout -b feature 603e667
Switched to a new branch 'feature'

DELL@DESKTOP-RCNKJC4 MINGW64 ~/Desktop/project-1/README.md (feature)
$ ls
README.md

DELL@DESKTOP-RCNKJC4 MINGW64 ~/Desktop/project-1/README.md (feature)
$ git log --oneline
603e667 (HEAD -> feature) Initial commit

DELL@DESKTOP-RCNKJC4 MINGW64 ~/Desktop/project-1/README.md (feature)
$ git checkout release
Switched to branch 'release'

DELL@DESKTOP-RCNKJC4 MINGW64 ~/Desktop/project-1/README.md (release)
$ ls
README.md  amendfile1  amendfile2  file1  file2  file3  revert.txt  wrongfile

DELL@DESKTOP-RCNKJC4 MINGW64 ~/Desktop/project-1/README.md (release)
$ git log --oneline
1784c8c (HEAD -> release) Revert "added again revert.txt"
7f89b62 deleted revert.txt
c992271 (origin/release) added revert.txt
327f01e first squash commit created file1
603e667 (feature) Initial commit

```

- `git log --oneline`
- `git checkout commit_id` (it will show the head at commit_id other are detached as will 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...

11. Git Hooks Challenge

a. Configure a pre-commit hook to reject commits without a message format (e.g., must start with JIRA-XXX).+

MINGW64:/c/Users/DELL/Desktop/project-1/README.md

```
DELL@DESKTOP-RCNKJ4 MINGW64 ~/Desktop/project-1/README.md (release)
$ cat .git/hooks/commit-msg
#!/bin/bash
# Commit message file passed by Git
commit_msg_file=$1
# Read first line of commit message
commit_msg=$(head -n1 "$commit_msg_file")
# Regex: Commit must start with JIRA- followed by numbers, e.g., JIRA-123
if ! [[ $commit_msg =~ ^JIRA-[0-9]+: ]]; then
    echo "X Commit rejected!
    Message must start with 'JIRA-XXX:'"
    exit 1
fi
exit 0

DELL@DESKTOP-RCNKJ4 MINGW64 ~/Desktop/project-1/README.md (release)
$ chmod 777 .git/hooks/commit-msg

DELL@DESKTOP-RCNKJ4 MINGW64 ~/Desktop/project-1/README.md (release)
$ git status
On branch release
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   file3
        new file:   hook.txt
        new file:   hook2.txt

DELL@DESKTOP-RCNKJ4 MINGW64 ~/Desktop/project-1/README.md (release)
$ git commit -m "added files"
X Commit rejected!
  Message must start with 'JIRA-XXX:'

DELL@DESKTOP-RCNKJ4 MINGW64 ~/Desktop/project-1/README.md (release)
$ git commit -m "JIRA-100:created file3 hook.txt & hook2.txt"
[release 97741a8] JIRA-100:created file3 hook.txt & hook2.txt
 3 files changed, 18 insertions(+)
 create mode 100644 hook.txt
 create mode 100644 hook2.txt

DELL@DESKTOP-RCNKJ4 MINGW64 ~/Desktop/project-1/README.md (release)
$ git log --oneline
97741a8 (HEAD -> release) JIRA-100:created file3 hook.txt & hook2.txt
1784c8c Revert "added again revert.txt"
7f89b62 deleted revert.txt
c992271 (origin/release) added revert.txt
327f01e first squash commit created file1
603e667 (feature) Initial commit
```

- vi .git/hooks/commit-msg

paste these code

```
#!/bin/bash
```

```
# Commit message file passed by Git
```

```
commit_msg_file=$1
```

Read first line of commit message

```
commit_msg=$(head -n1 "$commit_msg_file")
```

Regex: Commit must start with JIRA- followed by numbers, e.g., JIRA-123

```
if ! [[ $commit_msg =~ ^JIRA-[0-9]+: ]]; then
```

```
    echo "✗ Commit rejected!"
```

```
    echo "Message must start with 'JIRA-XXX:'"
```

```
    exit 1
```

```
fi
```

```
exit 0
```

-
- `chmod 777 .git/hooks/commit-msg`
 - then created and add one file the commit on it
 - it will show *commit rejected (commit should start with JIRA-XXX*
 - *commit again-----* `git commit -m "JIRA-101:created file in main"`
 - hence it will create a commit successfully

12. Squash Merge vs Rebase Merge

a. Show the difference between squash merge and rebase merge with evidence.

squash merge:

- squash all commits in release into single commit related to any single file
- using `git rebase -i --root` (an editor will open there replace pick with squash and save :wq!)

- again one more editor will get open then replace the commit message which you want to modify and save)
- `git log --oneline` (you will find only single commit all commits will be removed and squashed into single commit)
- `git checkout main` (switch to other branch)
- `ls` (check for files)
- `git log --oneline` (check for commits)
- `git merge --squash release` (you will merge all the files without commit and files will move to working area again you need to make a commit)
- `git commit -m "JIRA-101:created files through squash merge from release"`
- `git log --oneline` (hence you will find a single commit of all the files)
- squash merge is used to copy all files and from other branch without commits all merged files will be moved to tracking area where you need to make a commit and push it)
 - i. If the branch is shared: prefer `git merge` (no squash) or `git revert` for undoing changes. Don't rewrite (`rebase/force-push`) branches that teammates use.
 - ii. Squash merge is great for cleaning up many WIP commits into a single meaningful commit before landing into main.
 - iii. After `--squash`: delete or reset the source branch to avoid duplicate/duplicate-content merges later

rebase merge:

```

DELL@DESKTOP-RCNKJC4 MINGW64 ~/Desktop/project-1/README.md (main)
$ ls
README.md  file4

DELL@DESKTOP-RCNKJC4 MINGW64 ~/Desktop/project-1/README.md (main)
$ git log --oneline
e449286 (HEAD -> main) JIRA-101:added file4 in main
187a96d (origin/main, origin/HEAD) Initial commit

DELL@DESKTOP-RCNKJC4 MINGW64 ~/Desktop/project-1/README.md (main)
$ git rebase release
Successfully rebased and updated refs/heads/main.

DELL@DESKTOP-RCNKJC4 MINGW64 ~/Desktop/project-1/README.md (main)
$ ls
README.md  file3  file4  hookfile  hookfile2

DELL@DESKTOP-RCNKJC4 MINGW64 ~/Desktop/project-1/README.md (main)
$ git log --oneline
d98a11f (HEAD -> main) JIRA-101:added file4 in main
4c49596 (release) JIRA-101:created file3 in release
7db975e JIRA-101:created hookfile2 in release
b02d246 JIRA-101:created hookfile in main
187a96d (origin/main, origin/HEAD) Initial commit

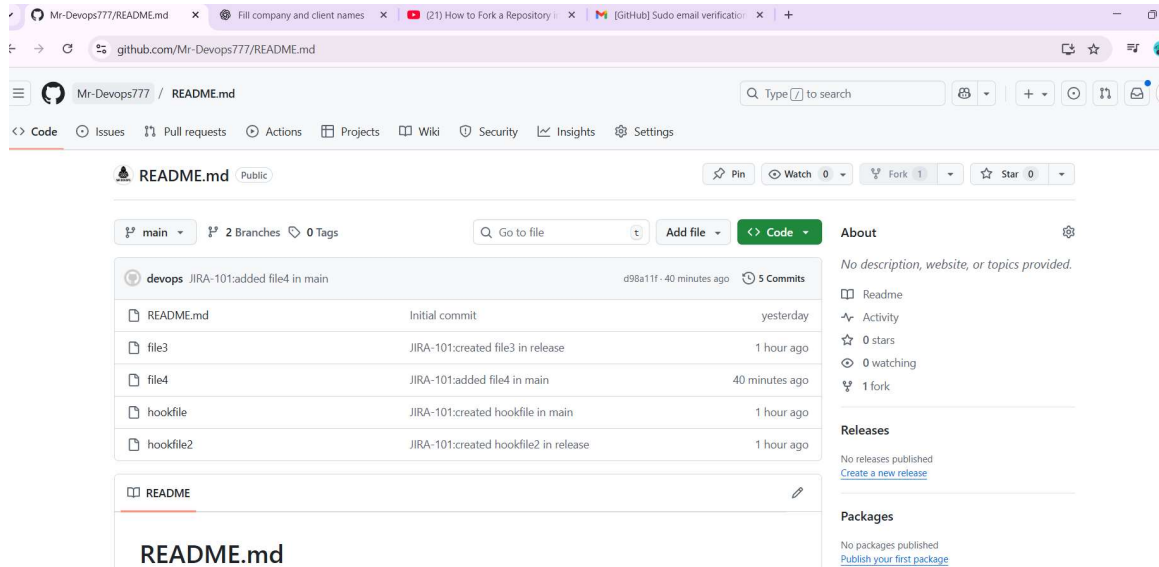
```

-
- git checkout main
- ls
- git log --oneline
- git rebase release
- ls
- git log --oneline
- *the rebase merge will copy all commits without overwriting a new commit it will save the commits as in release branch it keeps the history and dont create a new commit*
- *the basic differece between rebase merge,squash merge and merge*
- i. *reabse merge : merge all updates without creating new commit*
- ii. *squash merger :merge all updates without commit ids all files will be moved to tracking area where uou need to make a new commit and save*
- iii. *merge : it will merge all files and commits also created a new commit for merge*

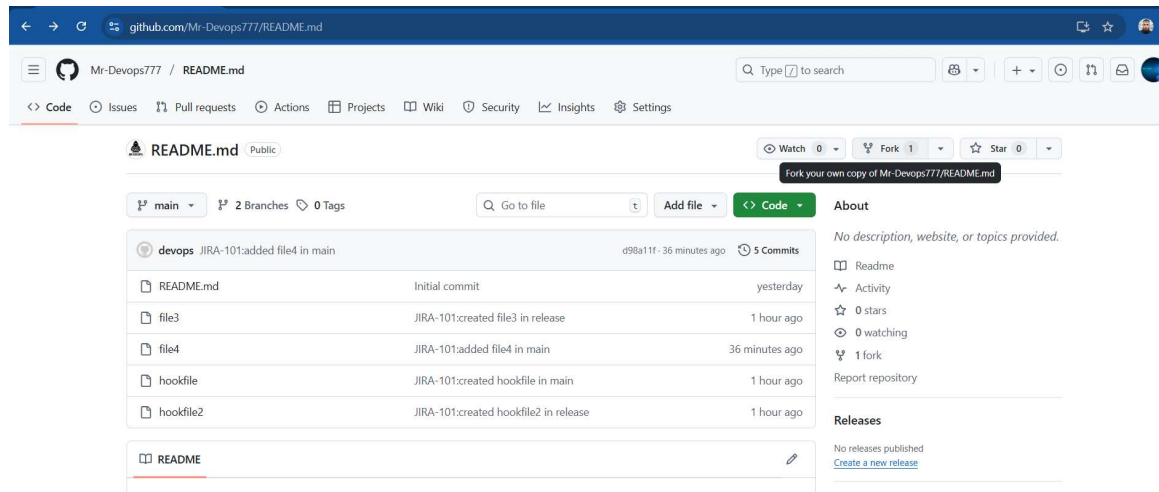
11. Fork & Pull Request Workflow

a. Fork a repo, make a change, and submit a pull request to the original repo.

account:1-(Mr-Devops-777)



account:2-(Mr-Devops-777)



- 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 account 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.

- pull request helps in merging the branches on git hub

12. Recover Lost Commit

a. Commit something, reset hard, and then recover it using git reflog.

MINGW64~/c/Users/DELL/Desktop/project-1/README.md

```
DELL@DESKTOP-RCNKJC4 MINGW64 ~/Desktop/project-1/README.md (main)
$ ls
README.md file3 file4 hookfile hookfile2

DELL@DESKTOP-RCNKJC4 MINGW64 ~/Desktop/project-1/README.md (main)
$ git log --oneline
d98a11f (HEAD -> main, upstream/main, origin/main, origin/HEAD) JIRA-101:added file4 in main
4c49596 (upstream/release, origin/release, release) JIRA-101:created file3 in release
7db975e JIRA-101:created hookfile2 in release
b02d246 JIRA-101:created hookfile in main
187a96d Initial commit

DELL@DESKTOP-RCNKJC4 MINGW64 ~/Desktop/project-1/README.md (main)
$ git reset --hard 187a96d
HEAD is now at 187a96d Initial commit

DELL@DESKTOP-RCNKJC4 MINGW64 ~/Desktop/project-1/README.md (main)
$ ls
README.md

DELL@DESKTOP-RCNKJC4 MINGW64 ~/Desktop/project-1/README.md (main)
$ git log --oneline
187a96d (HEAD -> main) Initial commit

DELL@DESKTOP-RCNKJC4 MINGW64 ~/Desktop/project-1/README.md (main)
$ git reset --hard d98a11f
HEAD is now at d98a11f JIRA-101:added file4 in main

DELL@DESKTOP-RCNKJC4 MINGW64 ~/Desktop/project-1/README.md (main)
$ ls
README.md file3 file4 hookfile hookfile2

DELL@DESKTOP-RCNKJC4 MINGW64 ~/Desktop/project-1/README.md (main)
$ git log --oneline
d98a11f (HEAD -> main, upstream/main, origin/main, origin/HEAD) JIRA-101:added file4 in main
4c49596 (upstream/release, origin/release, release) JIRA-101:created file3 in release
7db975e JIRA-101:created hookfile2 in release
b02d246 JIRA-101:created hookfile in main
187a96d Initial commit
```

-
- Is (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)
- Is (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)

- `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)