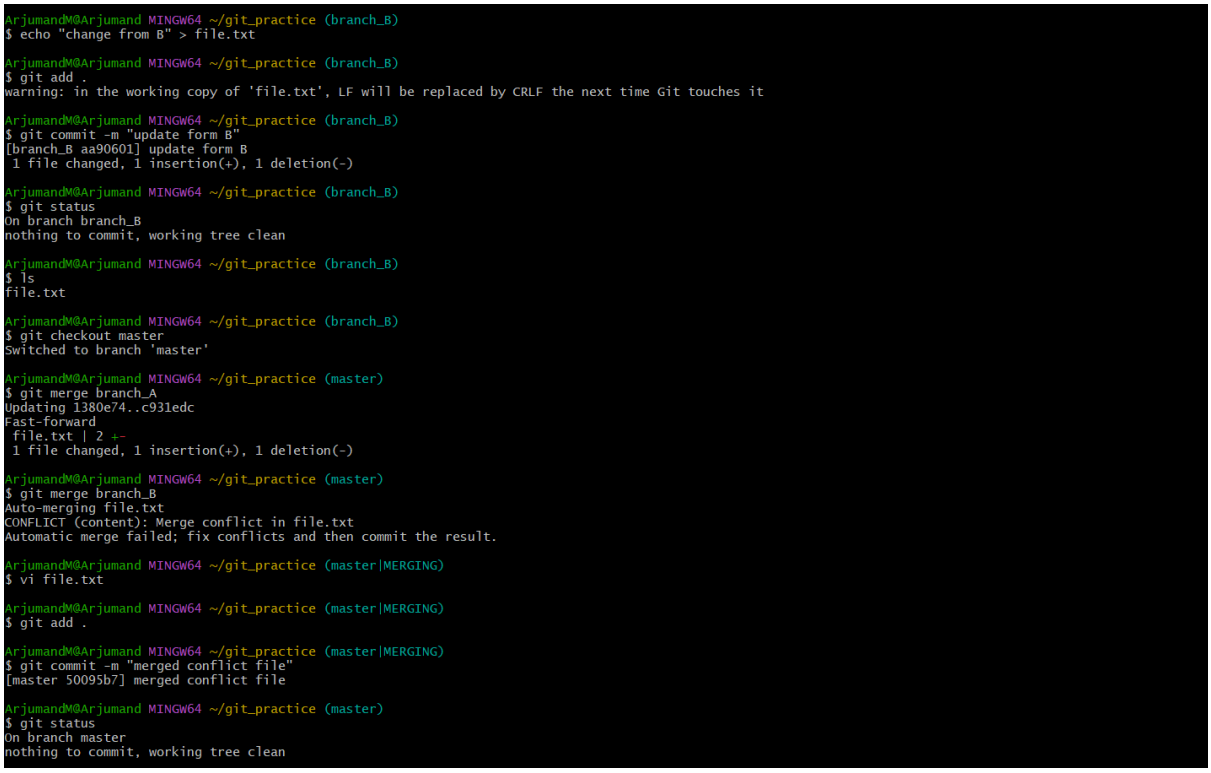
GIT & GIT HUB challenge

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

I have created a branch_A then then created a file file.txt int added file in tracking area



then I have created branch_B and then created a file with the same name file.txt and added to it to tracking area then switch to main branch and then started merging the branches by using command “git merge” first merge with with branch_A and then branch_B. then conflict created. I have entered into the file and change the content then added the file and added message then file got merged as you can in the below images.

```
ArjumandM@Arjumand MINGW64 ~/git_practice (master)
$ git branch
* master

ArjumandM@Arjumand MINGW64 ~/git_practice (master)
$ git checkout -b branch_A
Switched to a new branch 'branch_A'

ArjumandM@Arjumand MINGW64 ~/git_practice (branch_A)
$ echo "change from A" > file.txt

ArjumandM@Arjumand MINGW64 ~/git_practice (branch_A)
$ git status
On branch branch_A
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   file.txt

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

ArjumandM@Arjumand MINGW64 ~/git_practice (branch_A)
$ git add .
warning: in the working copy of 'file.txt', LF will be replaced by CRLF the next time Git touches it

ArjumandM@Arjumand MINGW64 ~/git_practice (branch_A)
$ git commit -m "update from A"
[branch_A c931edc] update from A
 1 file changed, 1 insertion(+), 1 deletion(-)

ArjumandM@Arjumand MINGW64 ~/git_practice (branch_A)
$ git stauts
git: 'stauts' is not a git command. See 'git --help'.

The most similar command is
        status

ArjumandM@Arjumand MINGW64 ~/git_practice (branch_A)
$ ls
file.txt

ArjumandM@Arjumand MINGW64 ~/git_practice (branch_A)
$ git checkout master
Switched to branch 'master'

ArjumandM@Arjumand MINGW64 ~/git_practice (master)
$ git checkout -b branch_B
Switched to a new branch 'branch_B'

ArjumandM@Arjumand MINGW64 ~/git_practice (branch_B)
$ echo "change from B" > file.txt

ArjumandM@Arjumand MINGW64 ~/git_practice (branch_B)
$ git add .
warning: in the working copy of 'file.txt', LF will be replaced by CRLF the next time Git touches it
```

2. Recover Deleted Branch

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

```
ArjumandM@Arjumand MINGW64 ~/git_practice (master)
$  git branch
  branch_A
  branch_B
* master

ArjumandM@Arjumand MINGW64 ~/git_practice (master)
$ git brnach -D branch_B
git: 'brnach' is not a git command. See 'git --help'.

The most similar command is
        branch

ArjumandM@Arjumand MINGW64 ~/git_practice (master)
$ git branch -D branch_B
Deleted branch branch_B (was aa90601).

ArjumandM@Arjumand MINGW64 ~/git_practice (master)
$ git branch
  branch_A
* master

ArjumandM@Arjumand MINGW64 ~/git_practice (master)
$ git reglog
git: 'reglog' is not a git command. See 'git --help'.

The most similar command is
        reflog

ArjumandM@Arjumand MINGW64 ~/git_practice (master)
$ git reflog
50095b7 (HEAD -> master) HEAD@{0}: commit (merge): merged conflict file
c931edc (branch_A) HEAD@{1}: merge branch_A: Fast-forward
1380e74 HEAD@{2}: checkout: moving from branch_B to master
aa90601 HEAD@{3}: commit: update form B
1380e74 HEAD@{4}: checkout: moving from master to branch_B
1380e74 HEAD@{5}: checkout: moving from branch_A to master
c931edc (branch_A) HEAD@{6}: commit: update from A
1380e74 HEAD@{7}: checkout: moving from master to branch_A
1380e74 HEAD@{8}: checkout: moving from branchB to master
398fc86 HEAD@{9}: checkout: moving from master to branchB
1380e74 HEAD@{10}: checkout: moving from branchA to master
1272748 HEAD@{11}: commit: update from A
1380e74 HEAD@{12}: checkout: moving from branchB to branchA
398fc86 HEAD@{13}: checkout: moving from master to branchB
1380e74 HEAD@{14}: merge branchA: Fast-forward
398fc86 HEAD@{15}: checkout: moving from branchA to master
1380e74 HEAD@{16}: commit: adding to check merge
398fc86 HEAD@{17}: checkout: moving from branchB to branchA
398fc86 HEAD@{18}: checkout: moving from master to branchB
398fc86 HEAD@{19}: checkout: moving from branchB to master
398fc86 HEAD@{20}: checkout: moving from master to branchB
398fc86 HEAD@{21}: commit (initial): first commit

ArjumandM@Arjumand MINGW64 ~/git_practice (master)
$  git log --oneline
50095b7 (HEAD -> master) merged conflict file
```

I have successfully recovered deleted branch by using commit ID. (git it from git log –oneline) or we can use command (git reflog)

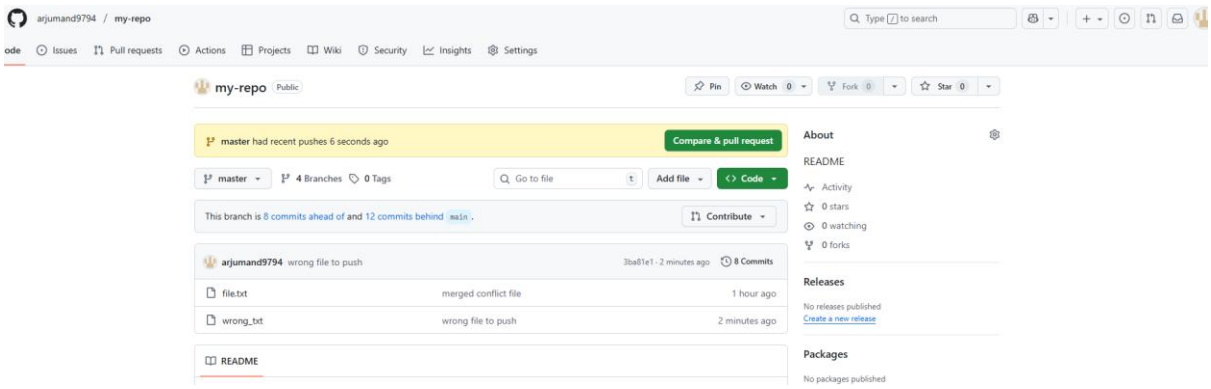
Either we use command git branch branch_B <commit Id> or we use git reset <commit Id>

```
ArjumandM@Arjumand MINGW64 ~/git_practice (master)
$  git log --oneline
50095b7 (HEAD -> master) merged conflict file
aa90601 update form B
c931edc (branch_A) update from A
1380e74 adding to check merge
398fc86 first commit

ArjumandM@Arjumand MINGW64 ~/git_practice (master)
$ git branch  branch_B aa90601

ArjumandM@Arjumand MINGW64 ~/git_practice (master)
$ git branch
  branch_A
  branch_B
* master

ArjumandM@Arjumand MINGW64 ~/git_practice (master)
$
```

3. **Undo Wrong Push**

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

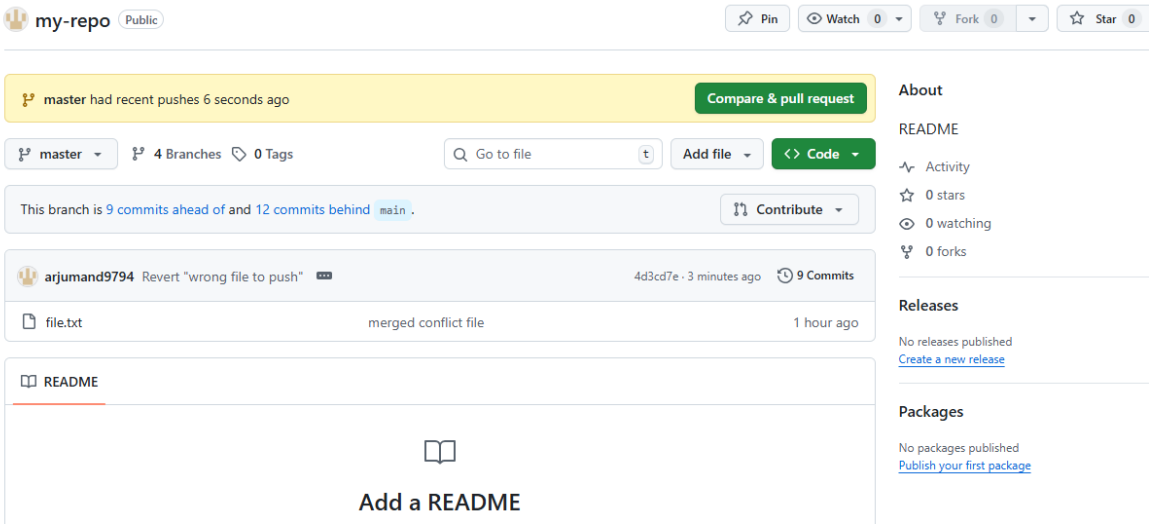
```
ArjumandM@Arjumand MINGW64 ~/git_practice (master)
$ echo "wrong file" > wrong_txt

ArjumandM@Arjumand MINGW64 ~/git_practice (master)
$ git add .
warning: in the working copy of 'wrong_txt', LF will be replaced by CRLF the next time Git touches it

ArjumandM@Arjumand MINGW64 ~/git_practice (master)
$ git commit -m "wrong file to push"
[master 3ba81e1] wrong file to push
 1 file changed, 1 insertion(+)
 create mode 100644 wrong_txt

ArjumandM@Arjumand MINGW64 ~/git_practice (master)
$ git status
On branch master
nothing to commit, working tree clean

ArjumandM@Arjumand MINGW64 ~/git_practice (master)
$ ls
file.txt  wrong_txt

ArjumandM@Arjumand MINGW64 ~/git_practice (master)
$ git push -u origin master
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 287 bytes | 287.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:arjumand9794/my-repo.git
   78c799c..3ba81e1  master -> master
branch 'master' set up to track 'origin/master'.
```



Wrong file is created and pushed to github repository successfully using command “git push

-u origin master” before pushing we need to connect to github repository by generating key and paste it in ssh key in github. Then connect remotely using ‘git remote -v’ then git add remote origin master “ssh key”.

Then push it then undo using ‘git resert <commit id> file got opened, just saved it then it got undo push successfully.



4. **Amend a Commit**

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

Using amend command I have give give same message for two file.

Fist I have created a file and added it and provided message then I have created another file then added it and used ‘git –amend’ then editor opened I simple saved it

```
ArjumandM@Arjumand MINGW64 ~/git_practice (master)
$ echo "Hello" > a.txt

ArjumandM@Arjumand MINGW64 ~/git_practice (master)
$ git add .
warning: in the working copy of 'a.txt', LF will be replaced by CRLF the next time Git touches it

ArjumandM@Arjumand MINGW64 ~/git_practice (master)
$ git commit -m "this is message for two files"
[master e2ffde7] this is message for two files
 1 file changed, 1 insertion(+)
 create mode 100644 a.txt

ArjumandM@Arjumand MINGW64 ~/git_practice (master)
$ echo " missing file" > b.txt

ArjumandM@Arjumand MINGW64 ~/git_practice (master)
$ git add b.txt
warning: in the working copy of 'b.txt', LF will be replaced by CRLF the next time Git touches it
```

```
ArjumandM@Arjumand MINGW64 ~/git_practice (master)
$ git commit --amend
[master 99a8176] this is message for two files
 Date: Sun Nov 16 17:20:54 2025 +0530
 2 files changed, 2 insertions(+)
 create mode 100644 a.txt
 create mode 100644 b.txt
```

```
# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
#
# Date:      Sun Nov 16 17:20:54 2025 +0530
#
# On branch master
# Your branch is ahead of 'origin/master' by 1 commit.
#   (use "git push" to publish your local commits)
#
# Changes to be committed:
#       new file:   a.txt
#       new file:   b.txt
#
~
```

5. **Cherry-pick a Commit**

a. Take a specific commit from one branch and apply it to another branch.

```
ArjumandM@Arjumand MINGW64 ~/git_practice (master)
$ git checkout branch_A
Switched to branch 'branch_A'

ArjumandM@Arjumand MINGW64 ~/git_practice (branch_A)
$ echo "hello to the world" wold.txt
hello to the world wold.txt

ArjumandM@Arjumand MINGW64 ~/git_practice (branch_A)
$ echo "hello to the world" > wold.txt

ArjumandM@Arjumand MINGW64 ~/git_practice (branch_A)
$ git add .
warning: in the working copy of 'wold.txt', LF will be replaced by CRLF the next time Git touches it

ArjumandM@Arjumand MINGW64 ~/git_practice (branch_A)
$ git commit -m "hello world message"
[branch_A b875f06] hello world message
 1 file changed, 1 insertion(+)
 create mode 100644 wold.txt

ArjumandM@Arjumand MINGW64 ~/git_practice (branch_A)
$ git checkout master
Switched to branch 'master'
Your branch is ahead of 'origin/master' by 1 commit.
  (use "git push" to publish your local commits)

ArjumandM@Arjumand MINGW64 ~/git_practice (master)
$ git log branch_A
commit b875f063c55968ea75d30affd2a39d61804ab030 (branch_A)
Author: Arjumand <Arjumand9794@gmail.com>
Date:   Sun Nov 16 17:53:51 2025 +0530
```

I have used cherry-pick to copy one commit id from branch_A branch. First of all I

havecreated one id called world.txt and switch to master user by using commid id of brach a I copied in master branch using command ‘git cherry-pick <commit id>’.

```
ArjumandM@Arjumand MINGW64 ~/git_practice (master)
$ git log branch_A
commit b875f063c55968ea75d30affd2a39d61804ab030 (branch_A)
Author: Arjumand <Arjumand9794@gmail.com>
Date:   Sun Nov 16 17:53:51 2025 +0530

    hello world message

commit c931edc9c475056001afbf4382fd1932f6e443b5
Author: Arjumand <Arjumand9794@gmail.com>
Date:   Sun Nov 16 15:22:38 2025 +0530

    update from A

commit 1380e74a76a7874976cb88b75aebe335296ebc57
Author: Arjumand <Arjumand9794@gmail.com>
Date:   Sun Nov 16 15:08:15 2025 +0530

    adding to check merge

commit 398fc86f67923d3bef30ab0fc998da0cb73e2408
Author: Arjumand <Arjumand9794@gmail.com>
Date:   Sun Nov 16 14:45:17 2025 +0530

    first commit

ArjumandM@Arjumand MINGW64 ~/git_practice (master)
$ git cherry pick b875f06
fatal: unknown commit pick

ArjumandM@Arjumand MINGW64 ~/git_practice (master)
$ git cherry-pick b875f06
[master d2b12d2] hello world message
 Date: Sun Nov 16 17:53:51 2025 +0530
 1 file changed, 1 insertion(+)
 create mode 100644 wold.txt

ArjumandM@Arjumand MINGW64 ~/git_practice (master)
$ ls
a.txt  b.txt  file.txt  wold.txt
```

6. **Interactive Rebase**

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

```
pick 4d3cd7e Revert "wrong file to push"
pick 69aba5f this is message for two files
pick d2b12d2 hello world message

# Rebase 3ba81e1..d2b12d2 onto 3ba81e1 (3 commands)
#
# Commands:
# p, pick <commit> = use commit
# r, reword <commit> = use commit, but edit the commit message
# e, edit <commit> = use commit, but stop for amending
# s, squash <commit> = use commit, but meld into previous commit
# f, fixup [-C | -c] <commit> = like "squash" but keep only the previous
#                    commit's log message, unless -C is used, in which case
#                    keep only this commit's message; -c is same as -C but
#                    opens the editor
# x, exec <command> = run command (the rest of the line) using shell
# b, break = stop here (continue rebase later with 'git rebase --continue')
# d, drop <commit> = remove commit
# l, label <label> = label current HEAD with a name
# t, reset <label> = reset HEAD to a label
# m, merge [-C <commit> | -c <commit>] <label> [# <oneline>]
# .       create a merge commit using the original merge commit's
# .       message (or the oneline, if no original merge commit was
# .       specified); use -c <commit> to reword the commit message
#
# These lines can be re-ordered; they are executed from top to bottom.
#
# If you remove a line here THAT COMMIT WILL BE LOST.
#
# However, if you remove everything, the rebase will be aborted.
#
```

```
pick 4d3cd7e Revert "wrong file to push"
squash 69aba5f this is message for two files
squash d2b12d2 hello world message

# Rebase 3ba81e1..d2b12d2 onto 3ba81e1 (3 commands)
#
# Commands:
# p, pick <commit> = use commit
# r, reword <commit> = use commit, but edit the commit message
# e, edit <commit> = use commit, but stop for amending
# s, squash <commit> = use commit, but meld into previous commit
# f, fixup [-C | -c] <commit> = like "squash" but keep only the previous
#                    commit's log message, unless -C is used, in which case
#                    keep only this commit's message; -c is same as -C but
#                    opens the editor
# x, exec <command> = run command (the rest of the line) using shell
# b, break = stop here (continue rebase later with 'git rebase --continue')
# d, drop <commit> = remove commit
# l, label <label> = label current HEAD with a name
# t, reset <label> = reset HEAD to a label
# m, merge [-C <commit> | -c <commit>] <label> [# <oneline>]
# .       create a merge commit using the original merge commit's
# .       message (or the oneline, if no original merge commit was
# .       specified); use -c <commit> to reword the commit message
#
# These lines can be re-ordered; they are executed from top to bottom.
#
# If you remove a line here THAT COMMIT WILL BE LOST.
#
# However, if you remove everything, the rebase will be aborted.
#
~
```

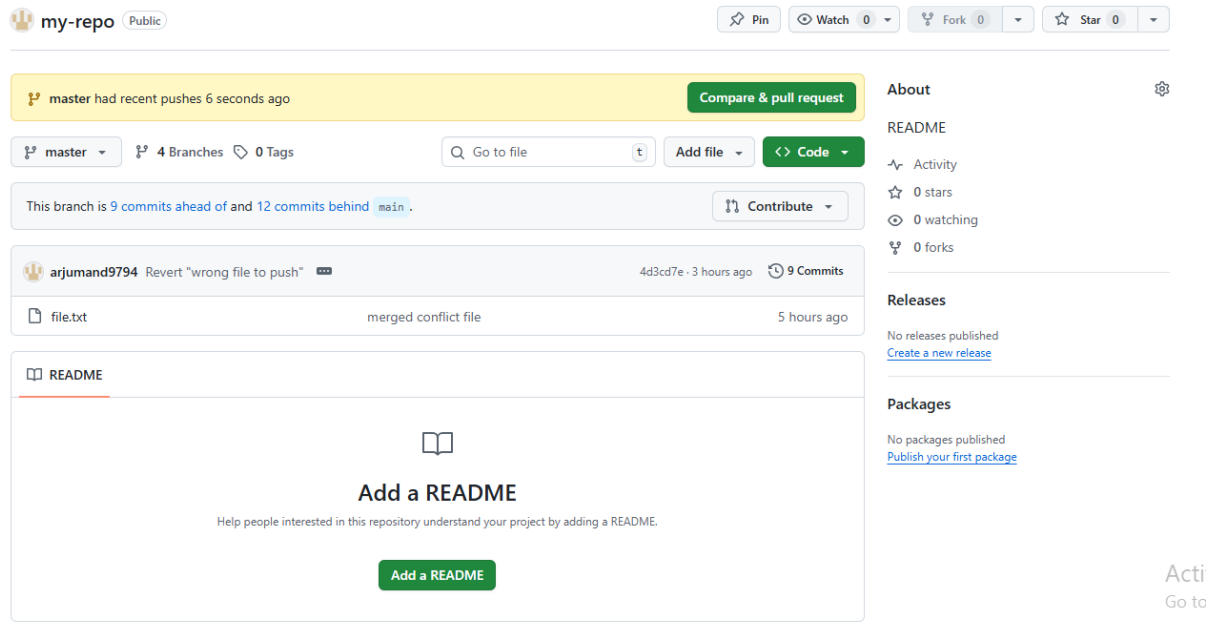
```
ArjumandM@Arjumand MINGW64 ~/git_practice (master)
$ git log --oneline
d2b12d2 (HEAD -> master) hello world message
69aba5f this is message for two files
4d3cd7e (origin/master) Revert "wrong file to push"
3ba81e1 wrong file to push
78c799c Revert "wrong file is created"
0961e2f wrong file is created
50095b7 merged conflict file
aa90601 (branch_B) update form B
c931edc update from A
1380e74 adding to check merge
398fc86 first commit

ArjumandM@Arjumand MINGW64 ~/git_practice (master)
$ git rebase -i HEAD~3
[detached HEAD c8f487a] Revert "wrong file to push"
 Date: Sun Nov 16 17:08:59 2025 +0530
 4 files changed, 3 insertions(+), 1 deletion(-)
 create mode 100644 a.txt
 create mode 100644 b.txt
 create mode 100644 wold.txt
 delete mode 100644 wrong_txt
Successfully rebased and updated refs/heads/master.

ArjumandM@Arjumand MINGW64 ~/git_practice (master)
$ git log --oneline
c8f487a (HEAD -> master) Revert "wrong file to push"
3ba81e1 wrong file to push
78c799c Revert "wrong file is created"
0961e2f wrong file is created
50095b7 merged conflict file
aa90601 (branch_B) update form B
c931edc update from A
1380e74 adding to check merge
398fc86 first commit
```

7. **Tagging & Release**

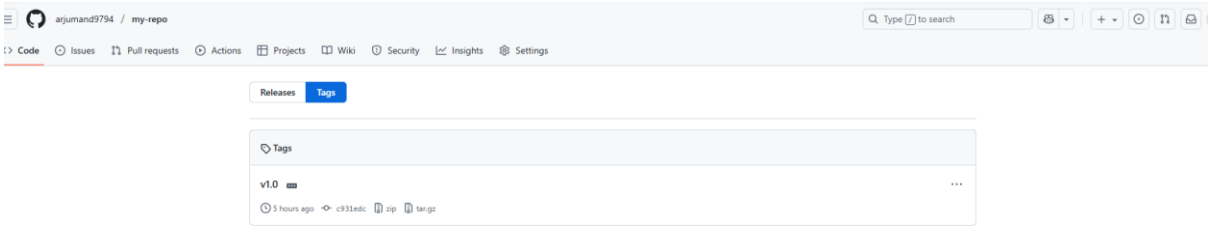
    a. Create a version tag (v1.0), push it to GitHub, then delete and restore it.

```
ArjumandM@Arjumand MINGW64 ~/git_practice (master|REBASE 1/3)
$ git tag v1.0

ArjumandM@Arjumand MINGW64 ~/git_practice (master|REBASE 1/3)
$ git push origin v1.0
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:arjumand9794/my-repo.git
 * [new tag]         v1.0 -> v1.0
```

Created v1.0 and pushed to github before refresh you can there is no tags.

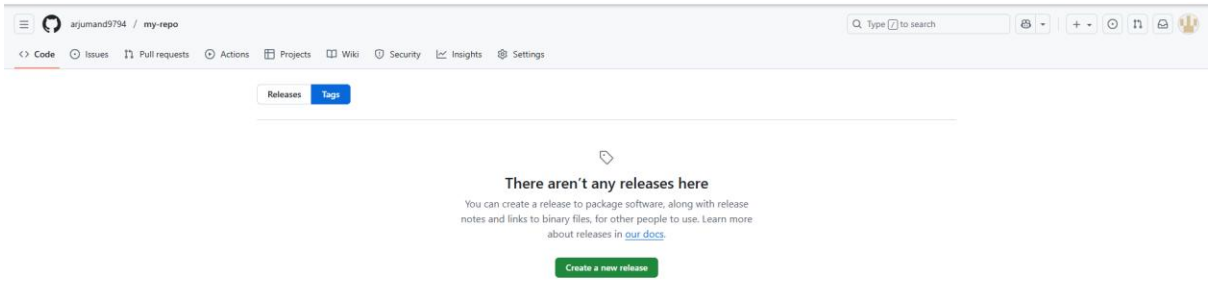


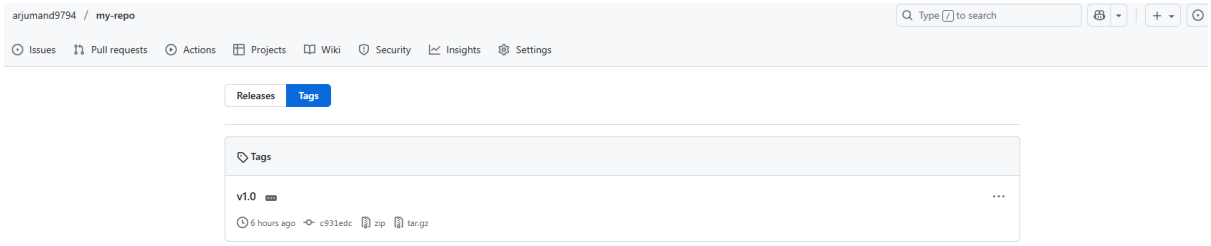
Now we can see the tag was pushed successfully



Now deleted version v1.0

No tags here in github



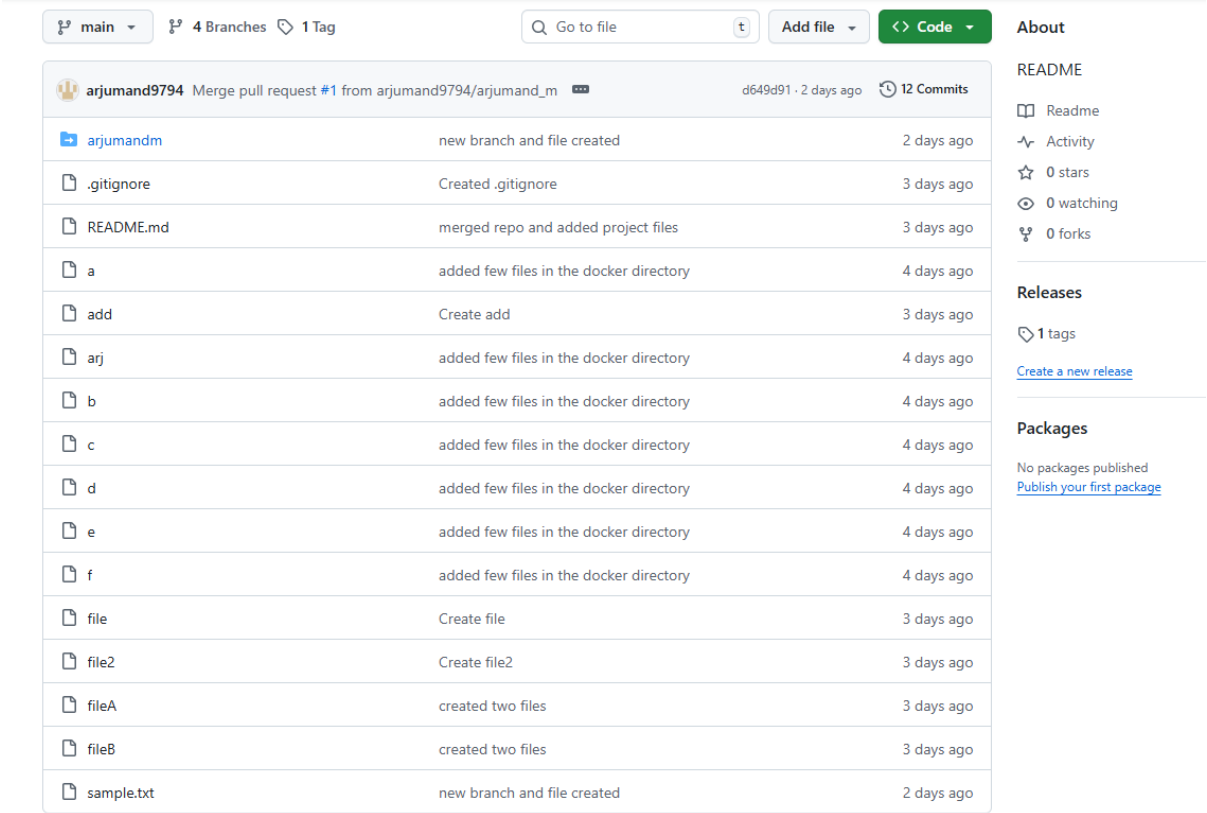



```
ArjumandM@Arjumand MINGW64 ~/git_practice (master|REBASE 1/3)
$ git tag -d v1.0
Deleted tag 'v1.0' (was c931edc)

ArjumandM@Arjumand MINGW64 ~/git_practice (master|REBASE 1/3)
$ git push origin :refs/tags/v1.0
To github.com:arjumand9794/my-repo.git
 - [deleted]         v1.0

ArjumandM@Arjumand MINGW64 ~/git_practice (master|REBASE 1/3)
$ git tag v1.0

ArjumandM@Arjumand MINGW64 ~/git_practice (master|REBASE 1/3)
$ git push origin v1.0
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:arjumand9794/my-repo.git
 * [new tag]         v1.0 -> v1.0
```

8. **Clone with Sparse Checkout**

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

```
ArjumandM@Arjumand MINGW64 ~/git_practice/my-repo (main|SPARSE)
$ git clone --no-checkout git@github.com:arjumand9794/my-repo.git
Cloning into 'my-repo'...
remote: Enumerating objects: 57, done.
remote: Counting objects: 100% (57/57), done.
remote: Compressing objects: 100% (30/30), done.
remote: Total 57 (delta 10), reused 43 (delta 7), pack-reused 0 (from 0)
Receiving objects: 100% (57/57), 8.90 KiB | 2.22 MiB/s, done.
Resolving deltas: 100% (10/10), done.

ArjumandM@Arjumand MINGW64 ~/git_practice/my-repo (main|SPARSE)
$ cd my-repo

ArjumandM@Arjumand MINGW64 ~/git_practice/my-repo/my-repo (main)
$ git sparse-checkout init --cone

ArjumandM@Arjumand MINGW64 ~/git_practice/my-repo/my-repo (main|SPARSE)
$ git sparse-checkout init --cone

ArjumandM@Arjumand MINGW64 ~/git_practice/my-repo/my-repo (main|SPARSE)
$ git sparse-checkout set arjumandm/

ArjumandM@Arjumand MINGW64 ~/git_practice/my-repo/my-repo (main|SPARSE)
$ git checkout main
Already on 'main'
Your branch is up to date with 'origin/main'.

ArjumandM@Arjumand MINGW64 ~/git_practice/my-repo/my-repo (main|SPARSE)
$ ls
README.md  a  add  arj  arjumandm/  b  c  d  e  f  file  file2  fileA  fileB  sample.txt
```



9. Reset vs Revert Challenge

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

- Reset example

1. git reset --hard B

What happens:

- Moves HEAD back to **B**
- **Deletes commit C** from history
- Working directory becomes exactly like commit B
- In below image we can see 58d1840 is comes first from the middle if use command ‘git rest 58d1840’.

```
ArjumandM@Arjumand MINGW64 ~/git_practice/my-repo/my-repo (main|SPARSE)
$ git log --oneline
d649d91 (HEAD -> main, origin/main, origin/HEAD) Merge pull request #1 from arjumand9794/arjumand_m
a9f08df new branch and file created
997245a (origin/Arjumand_mm) Merge branch 'main' of github.com:arjumand9794/my-repo
46ee45a created two files
58d1840 Create file2
200320e Create file
095984e merged repo and added project files
b6bed2e Created .gitignore
5c69f79 Initial commit
bb770b2 Create add
8d667d0 first commit
833ed75 added few files in the docker directory

ArjumandM@Arjumand MINGW64 ~/git_practice/my-repo/my-repo (main|SPARSE)
$ git reset --hard <hash_B>
bash: syntax error near unexpected token `newline'

ArjumandM@Arjumand MINGW64 ~/git_practice/my-repo/my-repo (main|SPARSE)
$ git reset --hard 58d1840
HEAD is now at 58d1840 Create file2

ArjumandM@Arjumand MINGW64 ~/git_practice/my-repo/my-repo (main|SPARSE)
$ git log --oneline
58d1840 (HEAD -> main) Create file2
200320e Create file
095984e merged repo and added project files
b6bed2e Created .gitignore
5c69f79 Initial commit
bb770b2 Create add
8d667d0 first commit
833ed75 added few files in the docker directory
```

 Creates a new commit **D** that undoes changes of C

 Does NOT delete C

```
ArjumandM@Arjumand MINGW64 ~/git_practice/my-repo/my-repo (main|SPARSE)
$ git log --oneline
58d1840 (HEAD -> main) Create file2
200320e Create file
095984e merged repo and added project files
b6bed2e Created .gitignore
5c69f79 Initial commit
bb770b2 Create add
8d667d0 first commit
833ed75 added few files in the docker directory

ArjumandM@Arjumand MINGW64 ~/git_practice/my-repo/my-repo (main|SPARSE)
$ git revert 200320e
[main 5f58241] Revert "Create file"
 1 file changed, 1 deletion(-)
 delete mode 100644 file

ArjumandM@Arjumand MINGW64 ~/git_practice/my-repo/my-repo (main|SPARSE)
$ git log --oneline
5f58241 (HEAD -> main) Revert "Create file"
58d1840 Create file2
200320e Create file
095984e merged repo and added project files
b6bed2e Created .gitignore
5c69f79 Initial commit
bb770b2 Create add
8d667d0 first commit
833ed75 added few files in the docker directory
```

10. **Detached HEAD Challenge**

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

```
ArjumandM@Arjumand MINGW64 ~/git_practice/my-repo/my-repo (main|SPARSE)
$ git checkout 095984e
Note: switching to '095984e'.

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 095984e merged repo and added project files

ArjumandM@Arjumand MINGW64 ~/git_practice/my-repo/my-repo ((095984e...)|SPARSE)
$ git checkout 095984e
HEAD is now at 095984e merged repo and added project files

ArjumandM@Arjumand MINGW64 ~/git_practice/my-repo/my-repo ((095984e...)|SPARSE)
$ git log --oneline
095984e (HEAD) merged repo and added project files
b6bed2e Created .gitignore
5c69f79 Initial commit
bb770b2 Create add
8d667d0 first commit
833ed75 added few files in the docker directory

ArjumandM@Arjumand MINGW64 ~/git_practice/my-repo/my-repo ((095984e...)|SPARSE)
$ git checkout -b branch_A
Switched to a new branch 'branch_A'
```

11. **Git Hooks Challenge**

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

```
#!/bin/bash

# Get commit message
commit_msg=$(cat "$1")

# Regex: must start with JIRA- followed by numbers
if [[ ! "$commit_msg" =~ ^JIRA-[0-9]+ ]]; then
  echo "Commit rejected!"
  echo " Commit message must start with: JIRA-XXX"
  exit 1
fi

exit 0

~
```

Create a pre-commit file in .git/hooks/ and add a script that checks if the commit message starts with JIRA-XXX.

Make it executable (chmod +x pre-commit); Git will now reject any commit whose message doesn't follow the required format.

```
ArjumandM@Arjumand MINGW64 ~/git_practice/my-repo/my-repo (branch_A|SPARSE)
$ cd .git/hooks

ArjumandM@Arjumand MINGW64 ~/git_practice/my-repo/my-repo/.git/hooks (GIT_DIR!|SPARSE)
$ pwd
/c/Users/ArjumandM/git_practice/my-repo/my-repo/.git/hooks

ArjumandM@Arjumand MINGW64 ~/git_practice/my-repo/my-repo/.git/hooks (GIT_DIR!|SPARSE)
$  vi pre-commit

ArjumandM@Arjumand MINGW64 ~/git_practice/my-repo/my-repo/.git/hooks (GIT_DIR!|SPARSE)
$ chmod +x pre-commit

ArjumandM@Arjumand MINGW64 ~/git_practice/my-repo/my-repo/.git/hooks (GIT_DIR!|SPARSE)
$ ./pre-commit
cat: '': No such file or directory
Commit rejected!
 Commit message must start with: JIRA-XXX

ArjumandM@Arjumand MINGW64 ~/git_practice/my-repo/my-repo/.git/hooks (GIT_DIR!|SPARSE)
$  vi pre-commit
```

12. **Squash Merge vs Rebase Merge**

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

Squash Merge (What we did)

Command used:

git merge --squash feature

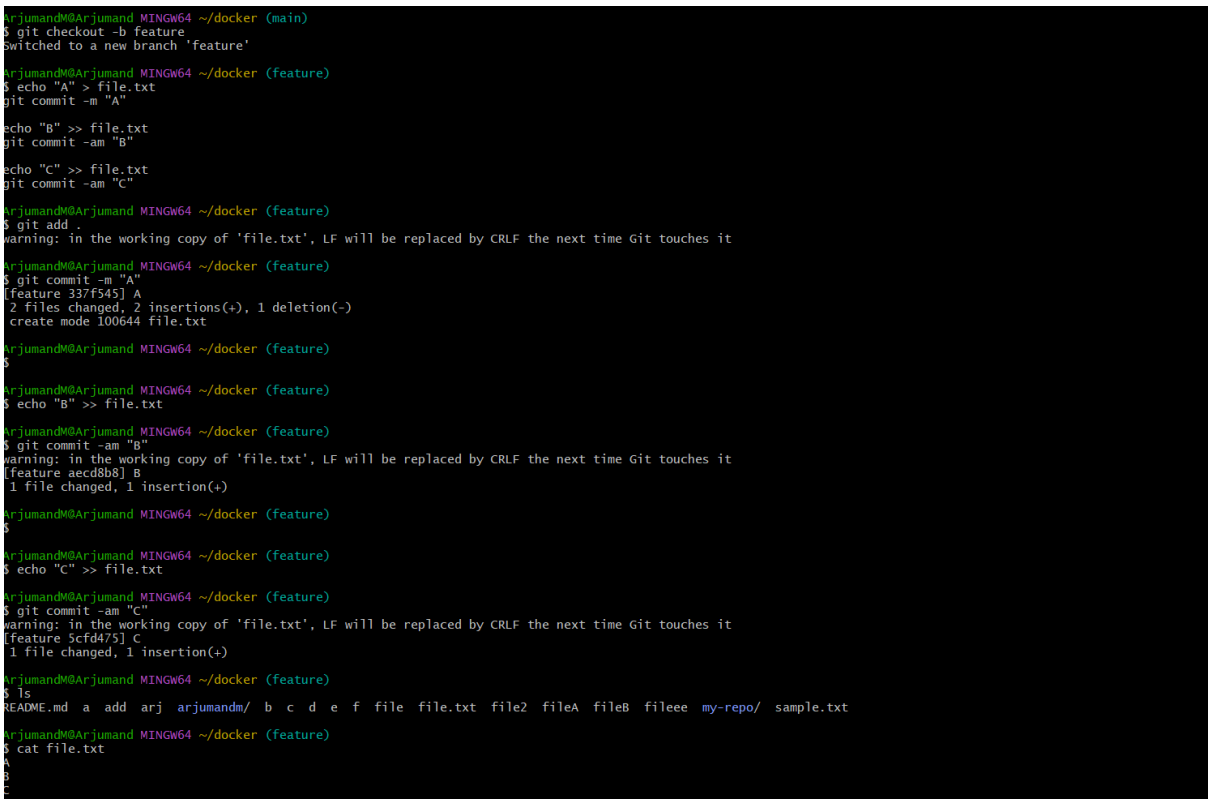
git commit -m "Squashed A B C"

What happened:

- All 3 commits (A, B, C) from feature were combined into **one single commit**.
- Main branch history shows **only 1 new commit**.
- Feature commit history is **not preserved**.

Evidence (your log):

HEAD -> main Squashed A B C ← 1 commit only



```
ArjumandM@Arjumand MINGW64 ~/docker (feature)
$ git checkout main
git commit -m "Squashed A B C"
Switched to branch 'main'
M       my-repo

ArjumandM@Arjumand MINGW64 ~/docker (main)
$ git merge --squash feature
Updating 4ba1588..5cfd475
Fast-forward
Squash commit -- not updating HEAD
 file.txt | 3 +++
 my-repo  | 2 +-
 2 files changed, 4 insertions(+), 1 deletion(-)
 create mode 100644 file.txt

ArjumandM@Arjumand MINGW64 ~/docker (main)
$ git commit -m "Squashed A B C"
[main fc5a000] Squashed A B C
 2 files changed, 4 insertions(+), 1 deletion(-)
 create mode 100644 file.txt

ArjumandM@Arjumand MINGW64 ~/docker (main)
$ git status
On branch main
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
  (commit or discard the untracked or modified content in submodules)
        modified:   my-repo (untracked content)

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

ArjumandM@Arjumand MINGW64 ~/docker (main)
$ git log --oneline
fc5a000 (HEAD -> main) Squashed A B C
4ba1588 Merge branch 'arjumand_m'
1375cf1 checking for merge
a9f08df new branch and file created
997245a (origin/main, origin/Arjumand_mm) Merge branch 'main' of github.com:arjumand9794/my-repo
46ee45a created two files
58d1840 Create file2
200320e Create file
095984e merged repo and added project files
b6bed2e (origin/master, master) Created .gitignore
5c69f79 Initial commit
bb770b2 Create add
8d667d0 first commit
833ed75 added few files in the docker directory

ArjumandM@Arjumand MINGW64 ~/docker (main)
$ git checkout feature
Switched to branch 'feature'
M       my-repo
```

Rebase Merge (What we did)

Command used:

git checkout feature

git rebase main

git checkout main

git merge feature # fast-forward

What happened:

- Commits A, B, C were **rebased and rewritten** as A’, B’, C’.
- Main branch got **all 3 commits individually**.
- History is linear and clean.
- No merge commit created (fast-forward).

Evidence (your log):

C'

B'

A'

(Each commit appears separately)

```
ArjumandM@Arjumand MINGW64 ~/docker (main)
$ git checkout feature
Switched to branch 'feature'
M       my-repo

ArjumandM@Arjumand MINGW64 ~/docker (feature)
$ echo "A" > reb.txt

echo "B" >> reb.txt
git commit -am "B"

echo "C" >> reb.txt
git commit -am "C"

ArjumandM@Arjumand MINGW64 ~/docker (feature)
$ git add .
warning: in the working copy of 'reb.txt', LF will be replaced by CRLF the next time Git touches it

ArjumandM@Arjumand MINGW64 ~/docker (feature)
$ git commit -m "A"
[feature 91b1bcb] A
 1 file changed, 1 insertion(+)
 create mode 100644 reb.txt

ArjumandM@Arjumand MINGW64 ~/docker (feature)
$

ArjumandM@Arjumand MINGW64 ~/docker (feature)
$ echo "B" >> reb.txt

ArjumandM@Arjumand MINGW64 ~/docker (feature)
$ git commit -am "B"
warning: in the working copy of 'reb.txt', LF will be replaced by CRLF the next time Git touches it
[feature 0c857cb] B
 1 file changed, 1 insertion(+)

ArjumandM@Arjumand MINGW64 ~/docker (feature)
$

ArjumandM@Arjumand MINGW64 ~/docker (feature)
$ echo "C" >> reb.txt

ArjumandM@Arjumand MINGW64 ~/docker (feature)
$ git commit -am "C"
warning: in the working copy of 'reb.txt', LF will be replaced by CRLF the next time Git touches it
[feature 959d02f] C
 1 file changed, 1 insertion(+)

ArjumandM@Arjumand MINGW64 ~/docker (feature)
$ git rebase main
Auto-merging file.txt
CONFLICT (add/add): Merge conflict in file.txt
error: could not apply 337f545... A
hint: Resolve all conflicts manually, mark them as resolved with
hint: "git add/rm <conflicted_files>", then run "git rebase --continue".
hint: You can instead skip this commit: run "git rebase --skip".
hint: To abort and get back to the state before "git rebase", run "git rebase --abort".
Could not apply 337f545... A
```



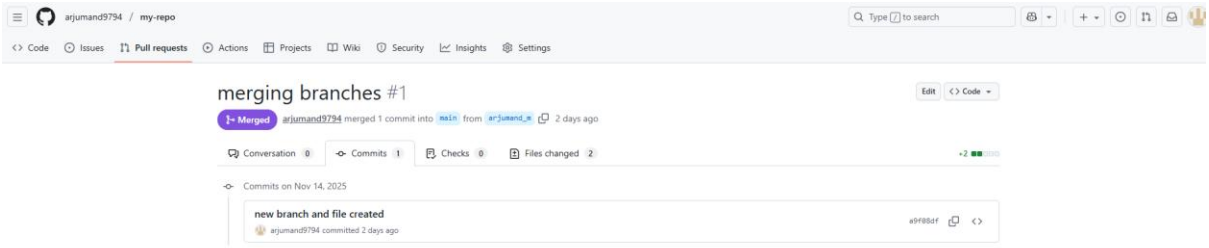
```
ArjumandM@Arjumand MINGW64 ~/docker (feature|REBASE 2/6)
$ git rebase --continue
file.txt: needs merge
You must edit all merge conflicts and then
mark them as resolved using git add

ArjumandM@Arjumand MINGW64 ~/docker (feature|REBASE 2/6)
$ vi file.txt

ArjumandM@Arjumand MINGW64 ~/docker (feature|REBASE 2/6)
$ git rebase --continue
file.txt: needs merge
You must edit all merge conflicts and then
mark them as resolved using git add

ArjumandM@Arjumand MINGW64 ~/docker (feature|REBASE 2/6)
$ vi file.txt

ArjumandM@Arjumand MINGW64 ~/docker (feature|REBASE 2/6)
$ git rebase main
fatal: It seems that there is already a rebase-merge directory, and
I wonder if you are in the middle of another rebase.  If that is the
case, please try
        git rebase (--continue | --abort | --skip)
If that is not the case, please
        rm -fr ".git/rebase-merge"
and run me again.  I am stopping in case you still have something
valuable there.
```

```
case, please try
        git rebase (--continue | --abort | --skip)
If that is not the case, please
        rm -fr ".git/rebase-merge"
and run me again.  I am stopping in case you still have something
valuable there.


ArjumandM@Arjumand MINGW64 ~/docker (feature|REBASE 2/6)
$ git rebase --continue
file.txt: needs merge
You must edit all merge conflicts and then
mark them as resolved using git add

ArjumandM@Arjumand MINGW64 ~/docker (feature|REBASE 2/6)
$ git add file.txt

ArjumandM@Arjumand MINGW64 ~/docker (feature|REBASE 2/6)
$ git rebase --continue
[detached HEAD 15f9fe2] B
 1 file changed, 1 insertion(+), 2 deletions(-)
Auto-merging file.txt
CONFLICT (content): Merge conflict in file.txt
error: could not apply 5cfd475... C
hint: Resolve all conflicts manually, mark them as resolved with
hint: "git add/rm <conflicted_files>", then run "git rebase --continue".
hint: You can instead skip this commit: run "git rebase --skip".
hint: To abort and get back to the state before "git rebase", run "git rebase --abort".
Could not apply 5cfd475... C

ArjumandM@Arjumand MINGW64 ~/docker (feature|REBASE 3/6)
$ vi file.txt

ArjumandM@Arjumand MINGW64 ~/docker (feature|REBASE 3/6)
$ git add file.txt

ArjumandM@Arjumand MINGW64 ~/docker (feature|REBASE 3/6)
$ git rebase --continue
[detached HEAD 14b7ef3] C
 1 file changed, 1 insertion(+), 1 deletion(-)
Successfully rebased and updated refs/heads/feature.

ArjumandM@Arjumand MINGW64 ~/docker (feature)
$ git checkout main
Switched to branch 'main'
M       my-repo

ArjumandM@Arjumand MINGW64 ~/docker (main)
$ Fast-forward
bash: Fast-forward: command not found

ArjumandM@Arjumand MINGW64 ~/docker (main)
$ git merge feature
Updating fc5a000..93d34c5
Fast-forward
 reb.txt | 3 +++
 1 file changed, 3 insertions(+)
 create mode 100644 reb.txt
```

13. Fork & Pull Request Workflow

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

I forked the original repository, cloned my fork, created a new branch, added a commit, pushed it, and then submitted a pull request to the original repository. The PR was reviewed and successfully merged into the main branch.



14. Recover Lost Commit

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

I made a commit, then removed it using git reset --hard.
The commit disappeared from history, but I found it in git reflog as 419cf71.
I restored it by running git cherry-pick 419cf71.

```
93d34c5 (HEAD -> main, feature) HEAD@{7}: rebase (continue) (finish): returning to refs/heads/feature
93d34c5 (HEAD -> main, feature) HEAD@{8}: rebase (continue) (pick): C
a1cd26a HEAD@{9}: rebase (continue) (pick): B
d2af3e4 HEAD@{10}: rebase (continue) (pick): A
14b7ef3 HEAD@{11}: rebase (continue): C
15f9fe2 HEAD@{12}: rebase (continue): B
d0afc20 HEAD@{13}: rebase (continue): A
fc5a000 HEAD@{14}: rebase (start): checkout main
959d02f HEAD@{15}: commit: C
0c857cb HEAD@{16}: commit: B
91b1bcb HEAD@{17}: commit: A
5cfd475 HEAD@{18}: checkout: moving from main to feature
fc5a000 HEAD@{19}: commit: Squashed A B C
4ba1588 HEAD@{20}: checkout: moving from feature to main
5cfd475 HEAD@{21}: commit: C
aecd8b8 HEAD@{22}: commit: B
337f545 HEAD@{23}: commit: A
4ba1588 HEAD@{24}: checkout: moving from main to feature
4ba1588 HEAD@{25}: checkout: moving from arjumand_m to main
32ca756 (origin/arjumand_m, arjumand_m) HEAD@{26}: commit: addded file
a9f08df HEAD@{27}: checkout: moving from main to arjumand_m
4ba1588 HEAD@{28}: merge arjumand_m: Merge made by the 'ort' strategy.
1375cf1 HEAD@{29}: checkout: moving from arjumand_m to main
a9f08df HEAD@{30}: checkout: moving from main to arjumand_m
1375cf1 HEAD@{31}: commit: checking for merge
997245a (origin/main, origin/Arjumand_mm) HEAD@{32}: checkout: moving from master to main
b6bed2e (origin/master, master) HEAD@{33}: checkout: moving from master to master
b6bed2e (origin/master, master) HEAD@{34}: checkout: moving from arjumand_m to master
a9f08df HEAD@{35}: commit: new branch and file created
997245a (origin/main, origin/Arjumand_mm) HEAD@{36}: checkout: moving from main to arjumand_m
997245a (origin/main, origin/Arjumand_mm) HEAD@{37}: pull origin main --allow-unrelated-histories: Merge made by the 'ort' strategy.
46ee45a HEAD@{38}: commit: created two files
095984e HEAD@{39}: commit (merge): merged repo and added project files
bb770b2 HEAD@{40}: Branch: renamed refs/heads/master to refs/heads/main
bb770b2 HEAD@{42}: pull: Fast-forward
8d667d0 HEAD@{43}: Branch: renamed refs/heads/main to refs/heads/master
8d667d0 HEAD@{45}: Branch: renamed refs/heads/main to refs/heads/main
8d667d0 HEAD@{47}: Branch: renamed refs/heads/master to refs/heads/main
8d667d0 HEAD@{49}: commit: first commit
833ed75 HEAD@{50}: commit (initial): added few files in the docker directory

ArjumandM@Arjumand MINGW64 ~/docker (main)
$ git cherry-pick 419cf71
[main a4b0b43] Lost commit
 Date: Mon Nov 17 00:25:02 2025 +0530
 1 file changed, 1 insertion(+)
 create mode 100644 lost.txt

ArjumandM@Arjumand MINGW64 ~/docker (main)
$ git log --oneline -n 5
a4b0b43 (HEAD -> main) Lost commit
93d34c5 (feature) C
a1cd26a B
d2af3e4 A
14b7ef3 C
```