Challenges of git and github

1) Resolve Merge Conflicts

```
[waseemakram@waseem myproject.git % vi file2
[waseemakram@waseem myproject.git % git add .
[waseemakram@waseem myproject.git % git commit -m "modified file2 "
 [release 908a7b4] modified file2
  1 file changed, 1 insertion(+)
[waseemakram@waseem myproject.git % cat file2
 hello world
```

Create a file named file2 and add lines as hello world
And
—>git add . (For adding all changes in tracking)
—> git commit -m “ “(for commit changes)
And then we view the file content where I have written “hello world “ in release branch file2

—> now I have to merge this file with same name file in other branch
We are merging this branch with main

```
[waseemakram@waseem myproject.git % git checkout main
 Switched to branch 'main'
 Your branch is ahead of 'origin/main' by 3 commits.
   (use "git push" to publish your local commits)
[waseemakram@waseem myproject.git % cat file2
 hello cat
```

Now we have to create a same name file from release branch as file2 and change the content of the file with “ hello cat”
—> now we should use same
—>git add.
—> git commit -m
And start merging the branch

—> git merge release (we merged the release branch with main)
We created merge conflict by this because we are having same named files with different content

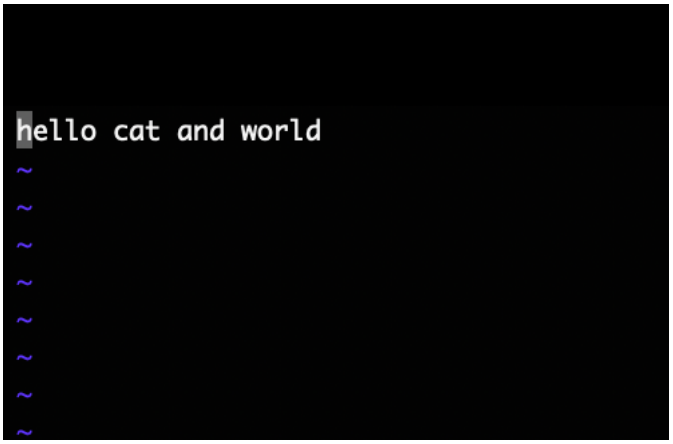
```
[waseemakram@waseem myproject.git % git merge release
Auto-merging file2
CONFLICT (content): Merge conflict in file2
Automatic merge failed; fix conflicts and then commit the result.
waseemakram@waseem myproject.git %
```

—> git status
We can see there we created a conflict on file2 as file2 from release has other content and file2 from main has other content
So it created a conflict
—-> to resolve this issue we can change the the content of file

```
waseemakram@waseem myproject.git % git status
On branch main
Your branch is ahead of 'origin/main' by 3 commits.
  (use "git push" to publish your local commits)

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")
waseemakram@waseem myproject.git %
```

```
<<<<<<< HEAD
hello cat
=======
hello world
>>>>>>> release
~
~
~
~
~
~
~
~
~
~
~
~
~
```

We changed the content of the file



And then
—> git add .
—> git commit -m
—> git push -u origin main

—> git merge will show you status of merge

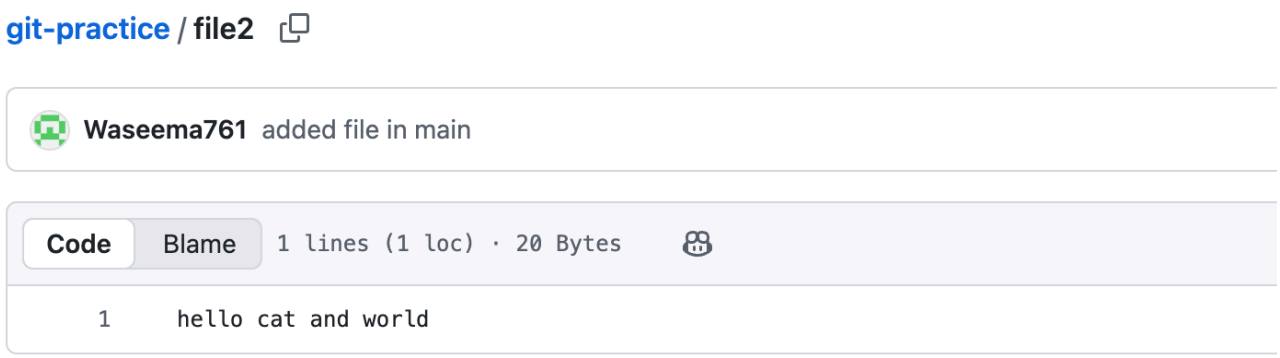
```
[waseemakram@waseem myproject.git % git merge
 Already up to date.
[waseemakram@waseem myproject.git % vi file2
```

And then we removed conflict of file2 by this

```
 nothing to commit, working tree clean
[waseemakram@waseem myproject.git % vi file2
[waseemakram@waseem myproject.git % cat file2
 hello cat and world
 waseemakram@waseem myproject.git %
```

```
[waseemakram@waseem myproject.git % git merge release
 Already up to date.
[waseemakram@waseem myproject.git % git push -u origin main
 Enumerating objects: 17, done.
 Counting objects: 100% (16/16), done.
 Delta compression using up to 8 threads
 Compressing objects: 100% (11/11), done.
 Writing objects: 100% (13/13), 1.22 KiB | 1.22 MiB/s, done.
 Total 13 (delta 5), reused 0 (delta 0), pack-reused 0 (from 0)
 remote: Resolving deltas: 100% (5/5), completed with 1 local object.
 To github.com:Waseema761/git-practice.git
    f45129d..fe6aee2  main -> main
 branch 'main' set up to track 'origin/main'.
 waseemakram@waseem myproject.git %
```

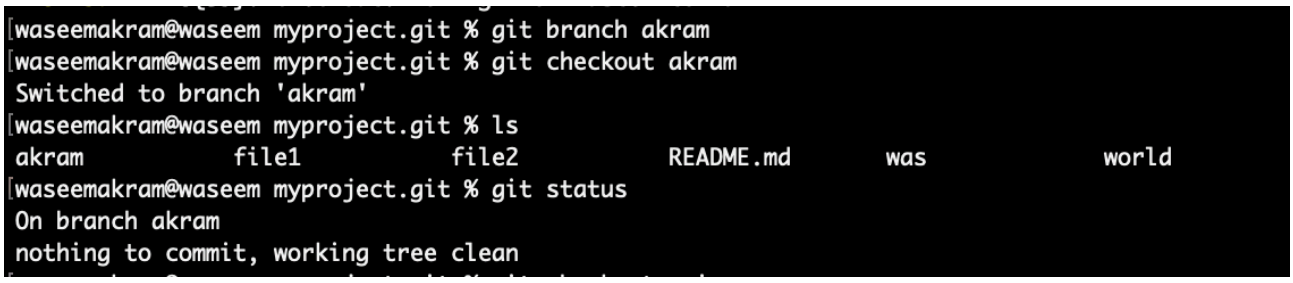
we pushed the changes to central repo



2)Recover Deleted Branch

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

—> git branch akram (created a branch name akram)
—> git checkout akram (switched to ram branch)



Now switch to main branch
—-> git branch -d akram (will delete the branch name akram)

We can see by reflog we can se fe6aee2 it is commit name of akram branch

```
waseemakram@waseem myproject.git % git reflog
fe6aee2 (HEAD -> main, origin/main, origin/HEAD) HEAD@{0}: checkout: moving from akram to main
fe6aee2 (HEAD -> main, origin/main, origin/HEAD) HEAD@{1}: checkout: moving from main to akram
fe6aee2 (HEAD -> main, origin/main, origin/HEAD) HEAD@{2}: commit: added file in main
1fa5df9 HEAD@{3}: commit (merge): added modified file in main
daa6e4e HEAD@{4}: checkout: moving from release to main
```

We then used—>git checkout -b akram fe6aee2
This will recover the deleted branch

```
waseemakram@waseem myproject.git % git checkout -b akram fe6aee2
Switched to a new branch 'akram'
waseemakram@waseem myproject.git % git branch
* akram
  main
  release
  waseem
waseemakram@waseem myproject.git % git log
commit fe6aee2382a084860a9a29fd1fb34866cd8db88e (HEAD -> akram, origin/main, origin/HEAD, main)
Author: waseem akram <waseem.akram082003@gmail.com>
```

3)Undo Wrong Push

We created a file name file3

```
waseemakram@waseem myproject.git % cat file3
wrong commit
waseemakram@waseem myproject.git % git status
On branch akram
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        file3

nothing added to commit but untracked files present (use "git add" to track)
waseemakram@waseem myproject.git % git add .
waseemakram@waseem myproject.git % git commit -m " wrong commit"
[akram 136f9c0]  wrong commit
 1 file changed, 1 insertion(+)
 create mode 100644 file3
waseemakram@waseem myproject.git % git push -u origin akram
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), 276 bytes | 276.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
```

—>And then we we git add .
—> And git commit -m “ wrong commit”
And later pushed it
—> Git push -u origin akram

```
[waseemakram@waseem myproject.git % git revert 136f9c
[akram 68113b7] Revert " wrong commit"
 1 file changed, 1 deletion(-)
 delete mode 100644 file3
waseemakram@waseem myproject.git %
```

```
commit 136f9c0554a21435dce8056a19aef1d7106322b0 (HEAD -> akram, origin/akram)
Author: waseem_akram <waseem.akram082003@gmail.com>
Date:   Mon Nov 17 13:58:26 2025 +0530

    wrong commit
```

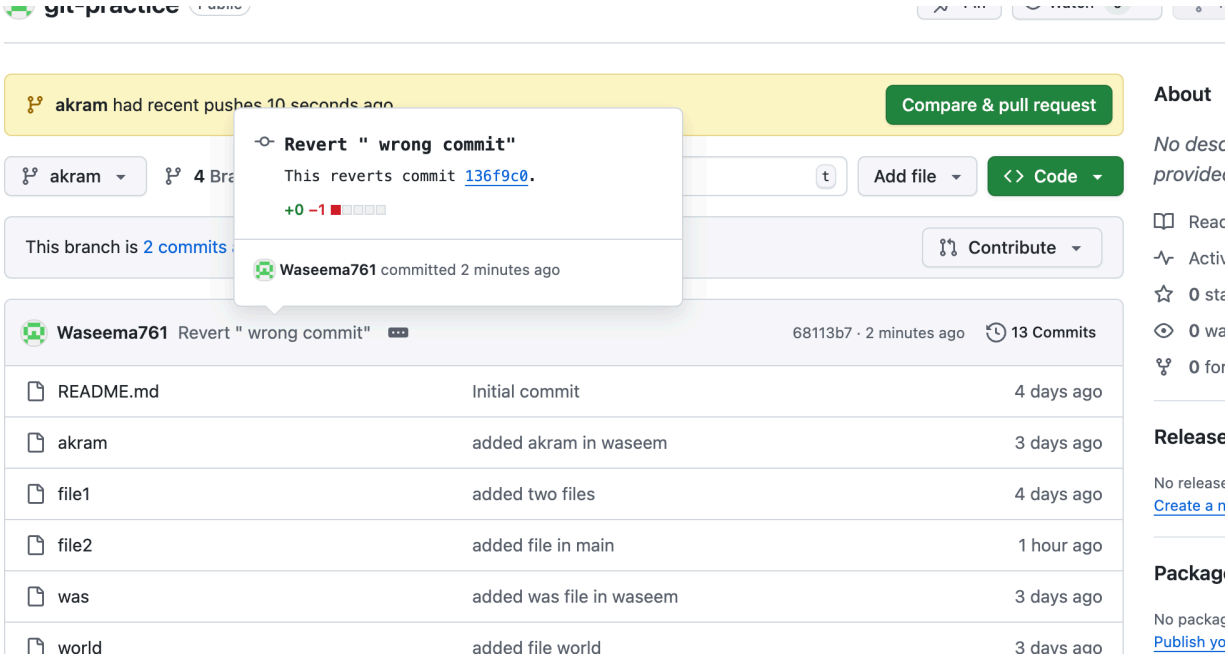
We copied the commit and git revert that commit after pushing

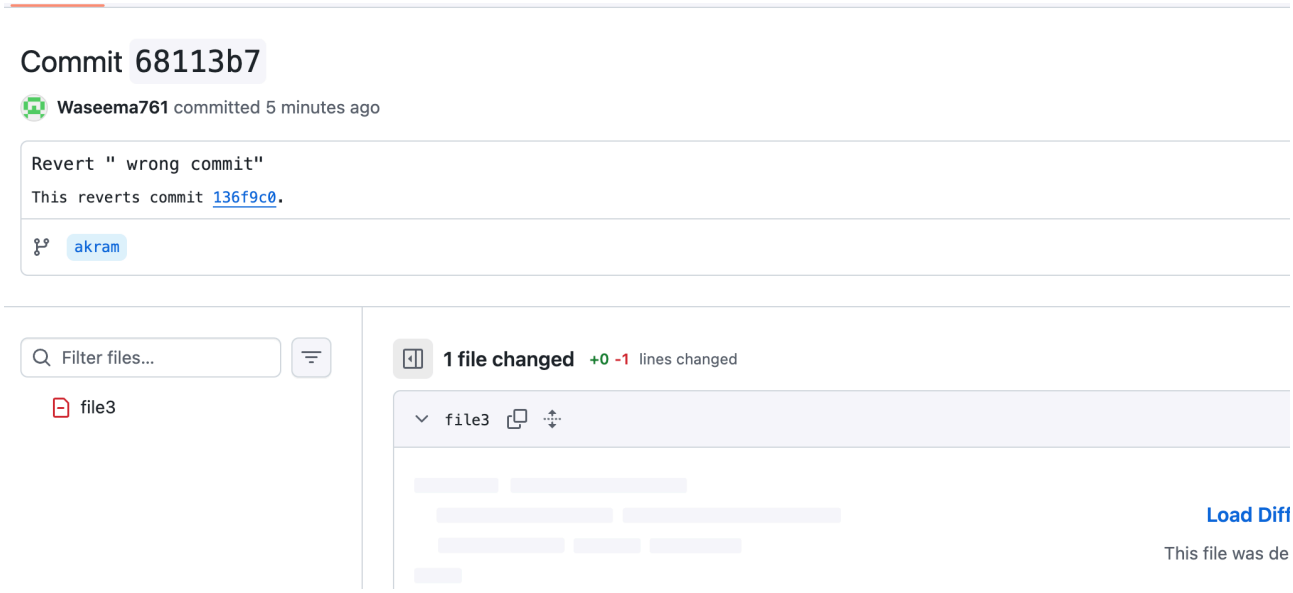
```
Revert " wrong commit"

This reverts commit 136f9c0554a21435dce8056a19aef1d7106322b0.

# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
#
# On branch akram
# Your branch is up to date with 'origin/akram'.
#
# Changes to be committed:
#       deleted:    file3
#
~
~
~
~
~
~
~
~
~
~
```

After save and exit we pushed to branch in central repo

And we git push
We can see revert “wrong commit”

```
waseemakram@waseem myproject.git % git revert 136f9c
[akram 68113b7] Revert " wrong commit"
 1 file changed, 1 deletion(-)
 delete mode 100644 file3
waseemakram@waseem myproject.git % git push
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (2/2), 251 bytes | 251.00 KiB/s, done.
Total 2 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To github.com:Waseema761/git-practice.git
   136f9c0..68113b7  akram -> akram
waseemakram@waseem myproject.git %
```





4) Amend a Commit

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

We created a file name file-a
And
—> git add .
—> Git commit -m
—> git push -u origin akram
We have created a file and pushed it too central repo

```
[waseemakram@waseem myproject.git % echo "hello" > file-a
[waseemakram@waseem myproject.git % git add .
[waseemakram@waseem myproject.git % git commit -m " added file-a"
 [akram 22712f4]  added file-a
  1 file changed, 1 insertion(+)
  create mode 100644 file-a
[waseemakram@waseem myproject.git % git push -u origin akram
 Enumerating objects: 7, done.
 Counting objects: 100% (7/7), done.
 Delta compression using up to 8 threads
 Compressing objects: 100% (5/5), done.
 Writing objects: 100% (6/6), 649 bytes | 649.00 KiB/s, done.
 Total 6 (delta 2), reused 0 (delta 0), pack-reused 0 (from 0)
 remote: Resolving deltas: 100% (2/2), completed with 1 local object.
 To github.com:Waseema761/git-practice.git
    68113b7..22712f4  akram -> akram
 branch 'akram' set up to track 'origin/akram'.
```

Now we have to create a file and add it but we should not commit that file

We created a file named file-b
—> git add .
And without commit we used
—> git commit —amend

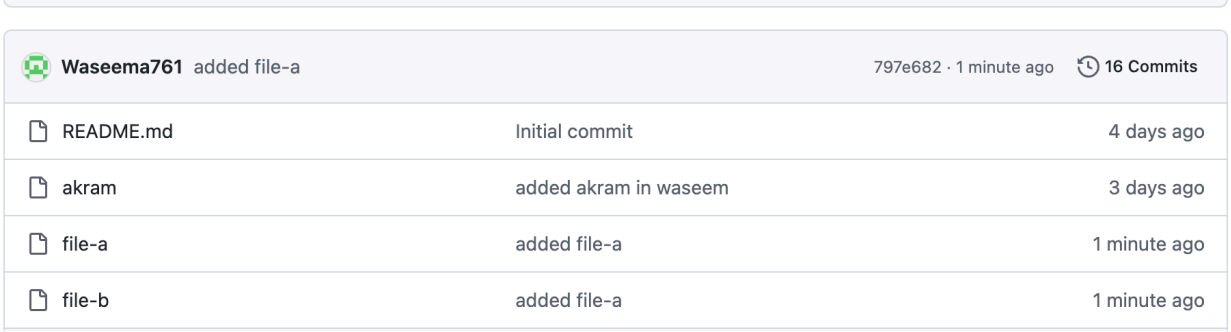
```
[waseemakram@waseem myproject.git % git log -1
 commit 22712f4f5c089ffb5ebcafd550e2c3047c16c919 (HEAD -> akram, origin/akram)
 Author: waseem_akram <waseem.akram082003@gmail.com>
 Date:   Mon Nov 17 14:32:31 2025 +0530

      added file-a
[waseemakram@waseem myproject.git % echo "missing file" > file-b
[waseemakram@waseem myproject.git % git add .
[waseemakram@waseem myproject.git % git commit --amend
 [akram 797e682]  added file-a
  Date: Mon Nov 17 14:32:31 2025 +0530
  2 files changed, 2 insertions(+)
  create mode 100644 file-a
  create mode 100644 file-b
 waseemakram@waseem myproject.git %
```

We can see two files having same commit

```
 added file-a

# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
#
# Date:      Mon Nov 17 14:32:31 2025 +0530
#
# On branch akram
# Your branch is up to date with 'origin/akram'.
#
# Changes to be committed:
#       new file:   file-a
#       new file:   file-b
#
~
~
~
~
~
~
~
~
~
```

Save and exit

And push force push the file




```
[waseemakram@waseem myproject.git % git commit --amend
 [akram 797e682]  added file-a
  Date: Mon Nov 17 14:32:31 2025 +0530
  2 files changed, 2 insertions(+)
  create mode 100644 file-a
  create mode 100644 file-b
[waseemakram@waseem myproject.git % git log -1
 commit 797e682d2f6061d9c63a2533d6fbdb9258e72ef6 (HEAD -> akram)
 Author: waseem_akram <waseem.akram082003@gmail.com>
 Date:   Mon Nov 17 14:32:31 2025 +0530

      added file-a
```

5)Cherry-pick a Commit

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

We are in a main branch
—> and checked its ls
And later switched to another branch
—> git checkout release
And ls

```
[waseemakram@waseem myproject.git % ls
 akram          file-a         file-b         file1          file2          README.md      was            wo
[waseemakram@waseem myproject.git % git checkout release
 Switched to branch 'release'
 Your branch is ahead of 'origin/release' by 1 commit.
   (use "git push" to publish your local commits)
[waseemakram@waseem myproject.git % ls
 file1          file2          README.md      world
```

Now we have to pick the file-a file-b files from main branch to release branch

```
[waseemakram@waseem myproject.git % git log -1
commit 797e682d2f6061d9c63a2533d6fbdb9258e72ef6 (HEAD -> akram, origin/akram)
Author: waseem_akram <waseem.akram082003@gmail.com>
Date:   Mon Nov 17 14:32:31 2025 +0530

    added file-a
```

We copied the commit of that file from git log

—> git cherry-pick 797e68 will pick that commit from main branch to release branch

```
[waseemakram@waseem myproject.git % git cherry-pick 797e68
[release b472d47]  added file-a
 Date: Mon Nov 17 14:32:31 2025 +0530
 2 files changed, 2 insertions(+)
 create mode 100644 file-a
 create mode 100644 file-b
[waseemakram@waseem myproject.git % ls
file-a          file-b          file1           file2           README.md       world
waseemakram@waseem myproject.git %
```

We can see file-a file-b two files are available in our release branch

6) Interactive Rebase

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

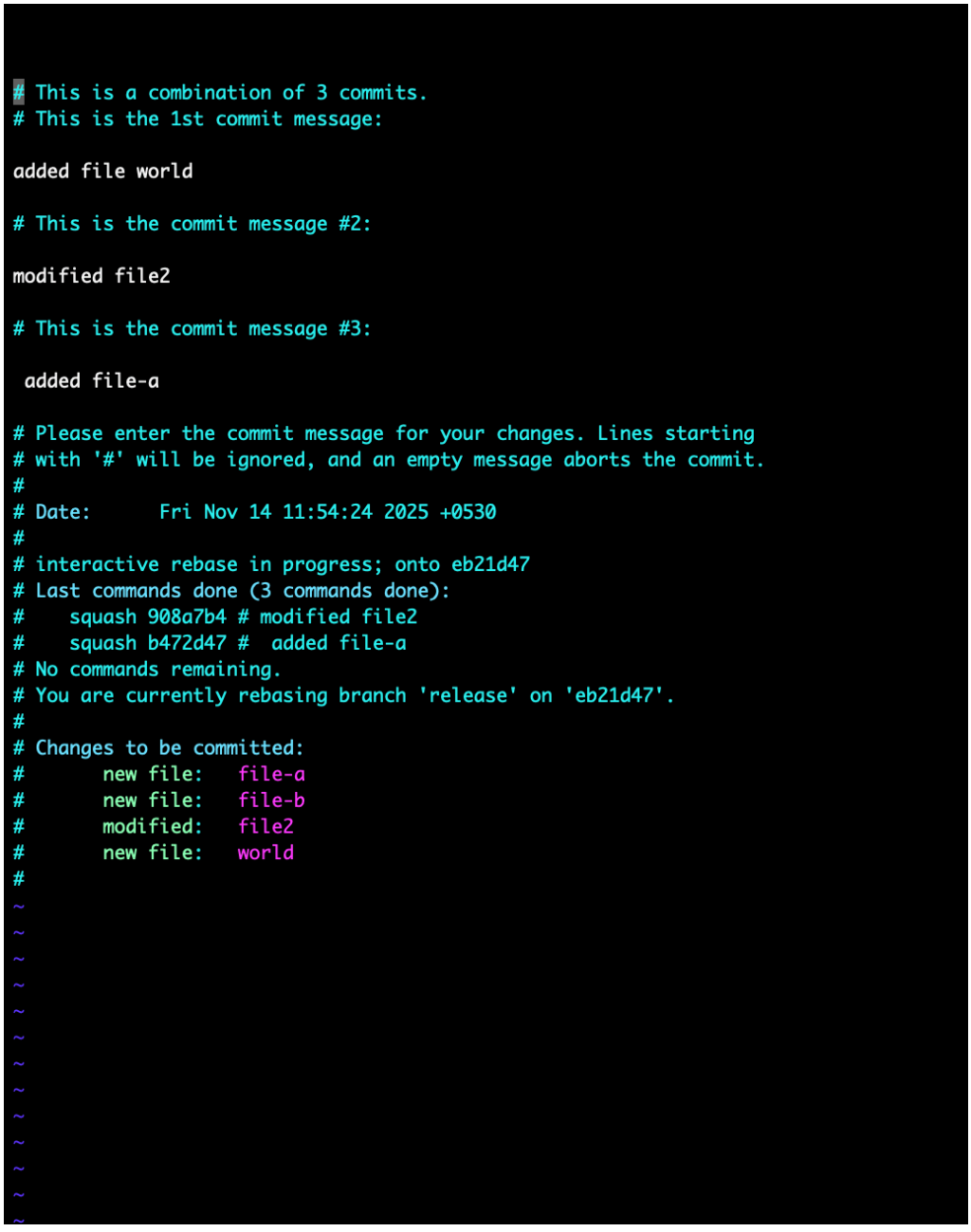
—> git rebase -i HEAD~3 (This will open the commit)

```
[waseemakram@waseem myproject.git % git rebase -i HEAD~3
[detached HEAD 0a77778] added file world
 Date: Fri Nov 14 11:54:24 2025 +0530
 4 files changed, 4 insertions(+)
 create mode 100644 file-a
 create mode 100644 file-b
 create mode 100644 world
Successfully rebased and updated refs/heads/release.
[waseemakram@waseem myproject.git % git log
```

—> now remove replace pick with squash And save and enter

```
pick 3266632 # added file world
squash 908a7b4 # modified file2
squash b472d47 #  added file-a

# Rebase eb21d47..b472d47 onto eb21d47 (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
# u, update-ref <ref> = track a placeholder for the <ref> to be updated
#                       to this position in the new commits. The <ref> is
#                       updated at the end of the rebase
#
# 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.
#
~
```

After save and enter again save and enter



—> Now check the git log

Before squash

```
waseemakram@waseem myproject.git % git log
commit b472d47efc29d2e33df7362d835509196a0d3ad8 (HEAD -> release)
Author: waseem_akram <waseem.akram082003@gmail.com>
Date:   Mon Nov 17 14:32:31 2025 +0530

     added file-a

commit 908a7b4cd3036bb3a8a13bc49ba3dde9c509239f
Author: waseem_akram <waseem.akram082003@gmail.com>
Date:   Mon Nov 17 12:15:15 2025 +0530

    modified file2

commit 326663239e3f500fc6566a8ee6e15f215581bf9d (origin/release)
Author: waseem_akram <waseem.akram082003@gmail.com>
Date:   Fri Nov 14 11:54:24 2025 +0530

    added file world

commit eb21d47498081e9a52550e508fe7ccd880645089
Author: waseem_akram <waseem.akram082003@gmail.com>
Date:   Thu Nov 13 19:07:19 2025 +0530

     added two files

commit 5c484c43990d928ee67a60f08abecc8204b1beaf
Author: Waseema761 <waseem.akram082003@gmail.com>
Date:   Thu Nov 13 19:00:06 2025 +0530
```

After squash

```
commit 0a777788f6b3eec6725e6ec826370b6c97bf9bba (HEAD -> release)
Author: waseem_akram <waseem.akram082003@gmail.com>
Date:   Fri Nov 14 11:54:24 2025 +0530

    added file world

    modified file2

     added file-a

commit eb21d47498081e9a52550e508fe7ccd880645089
Author: waseem_akram <waseem.akram082003@gmail.com>
Date:   Thu Nov 13 19:07:19 2025 +0530

     added two files
```

7) Tagging & Release

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

```
waseemakram@waseem myproject.git % git log -1
commit 8e4e35651ef7ac0eb7119803f6ee6a1bd239addc (HEAD -> release)
Author: waseem_akram <waseem.akram082003@gmail.com>
Date:   Mon Nov 17 16:13:13 2025 +0530

    added hello
waseemakram@waseem myproject.git % git tag 8e4e35 v1.0

fatal: Failed to resolve 'v1.0' as a valid ref.
waseemakram@waseem myproject.git % git tag v1.0
waseemakram@waseem myproject.git % git log -1
commit 8e4e35651ef7ac0eb7119803f6ee6a1bd239addc (HEAD -> release, tag: v1.0)
Author: waseem_akram <waseem.akram082003@gmail.com>
Date:   Mon Nov 17 16:13:13 2025 +0530

    added hello
waseemakram@waseem myproject.git %
```

—> Git tag =Lists all tags in the repository
Git tag 8e4e35 v1.0 we created a tag for commit

—> git push origin v1.0 will push the v1.0 (tag) commit

```
    added hello
waseemakram@waseem myproject.git % git push origin v1.0

Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 8 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (7/7), 619 bytes | 619.00 KiB/s, done.
Total 7 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), done.
To github.com:Waseema761/git-practice.git
 * [new tag]         v1.0 -> v1.0
waseemakram@waseem myproject.git % git push origin v1.0

Everything up-to-date
waseemakram@waseem myproject.git % git tag
```

—> Git tag -d v1.0 (will delete the tag)

```
waseemakram@waseem myproject.git % git tag -d v1.0

Deleted tag 'v1.0' (was 8e4e356)
waseemakram@waseem myproject.git % git tag

waseemakram@waseem myproject.git %
```

—> git tag (will show tag list .as tag has been deleted we are unavailable to view any tag)

```
waseemakram@waseem myproject.git % git push origin --delete v1.0

To github.com:Waseema761/git-practice.git
 - [deleted]         v1.0
waseemakram@waseem myproject.git % git tag -a v1.0 -m "restored v1.0 tag" 8e4e356

waseemakram@waseem myproject.git % git tag

v1.0
waseemakram@waseem myproject.git %
```

—> git tag -a v1.0 -m “ restored v1.0 tag “ 8e4e356
means you are **creating an annotated tag** named **v1.0** and adding a **message** to it.

—> We can simple use —> git tag v1.0 8e4e356
To restore the tag

8) Clone with Sparse Checkout

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

Created a clone of repo with sparse-test directory
Created a directory in main branch

```
waseemakram@waseem myproject.git % git clone --no-checkout git@github.com:Waseema761/git-practice.git sparse-test
cd sparse-test

Cloning into 'sparse-test'...
remote: Enumerating objects: 48, done.
remote: Counting objects: 100% (48/48), done.
remote: Compressing objects: 100% (27/27), done.
remote: Total 48 (delta 20), reused 35 (delta 11), pack-reused 0 (from 0)
Receiving objects: 100% (48/48), 5.43 KiB | 1.08 MiB/s, done.
Resolving deltas: 100% (20/20), done.
waseemakram@waseem sparse-test % git sparse-checkout init
```

```
waseemakram@waseem sparse-test % mkdir folder1
waseemakram@waseem sparse-test % git add .
waseemakram@waseem sparse-test % git commit -m " added folder 1"
On branch main
Your branch is up to date with 'origin/main'.

You are in a sparse checkout with 100% of tracked files present.

nothing to commit, working tree clean
waseemakram@waseem sparse-test % ls
akram           file1           file2           folder1         README.md       was             world
```

Git sparse-checkout init —cone (Enable sparse checkout)

```
waseemakram@waseem folder1 % git sparse-checkout init --cone
waseemakram@waseem folder1 % git sparse-checkout
error: need a subcommand
usage: git sparse-checkout (init | list | set | add | reapply | disable | check-rules) [<options>]

waseemakram@waseem folder1 % git sparse-checkout set folder1
waseemakram@waseem folder1 % git checkout main
Already on 'main'
Your branch is up to date with 'origin/main'.
waseemakram@waseem folder1 % git branch
* main
  release
```

Git sparse-checkout set folder1
(Select only the folder you want)

Now check the branch release
And check ls
We can see folder1 named directory is present

```
[waseemakram@waseem sparse-test % ls
 akram          file1          file2          folder1        README.md      was            world
[waseemakram@waseem sparse-test % git checkout release
 Switched to branch 'release'
 Your branch is up to date with 'origin/release'.
[waseemakram@waseem sparse-test % ls
 file1          file2          folder1        README.md      world
 waseemakram@waseem sparse-test %
```

9) Reset vs Revert Challenge

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

Created a new commit and used
—> git reset —hard 797e682 (to remove present head commit)

```
waseemakram@waseem myproject.git % git log --oneline

3ee1098 (HEAD -> akram)  add all files
797e682 (origin/akram)  added file-a
5f24119 Revert "added a file"
814bb26 added a file
68113b7 Revert " wrong commit"
136f9c0  wrong commit
fe6aee2 (origin/main, origin/HEAD, main)  added file in main
1fa5df9 added modified file in main
908a7b4 modified file2
daa6e4e  modefied the file2
e024cce Merge branch 'release'
f45129d Merge pull request #1 from Waseema761/waseem
a825ab0  added akram in waseem
3b76fff  added was file in waseem
3266632 (origin/release) added file world
eb21d47  added two files
5c484c4 Initial commit
[waseemakram@waseem myproject.git % git reset --hard 797e682
HEAD is now at 797e682  added file-a
```

—> the commit has been deleted (3ee1098)

```
[waseemakram@waseem myproject.git % git reset --hard 797e682
HEAD is now at 797e682  added file-a
waseemakram@waseem myproject.git % git log --oneline

797e682 (HEAD -> akram, origin/akram)  added file-a
5f24119 Revert "added a file"
814bb26 added a file
68113b7 Revert " wrong commit"
136f9c0  wrong commit
fe6aee2 (origin/main, origin/HEAD, main)  added file in main
1fa5df9 added modified file in main
908a7b4 modified file2
daa6e4e  modefied the file2
e024cce Merge branch 'release'
f45129d Merge pull request #1 from Waseema761/waseem
a825ab0  added akram in waseem
3b76fff  added was file in waseem
3266632 (origin/release) added file world
eb21d47  added two files
5c484c4 Initial commit
waseemakram@waseem myproject.git %
```

revert
—>. Git revert

Created a file and add . And commit it

```
waseemakram@waseem myproject.git % touch abc
waseemakram@waseem myproject.git % git add .
waseemakram@waseem myproject.git % git commit -m "added abc"
[akram 0d8cbef] added abc
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 abc
waseemakram@waseem myproject.git % git log --oneline
0d8cbef (HEAD -> akram) added abc
797e682 (origin/akram)  added file-a
5f24119 Revert "added a file"
814bb26 added a file
68113b7 Revert " wrong commit"
136f9c0  wrong commit
fe6aee2 (origin/main, origin/HEAD, main)  added file in main
```

Now use—> git revert 0d8cbef (this will revert the commit)

```
Revert "added abc"

This reverts commit 0d8cbefe0ffab85248ec7ff373dc925e32438be6.

# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
#
# On branch akram
# Your branch is ahead of 'origin/akram' by 1 commit.
#   (use "git push" to publish your local commits)
#
# Changes to be committed:
#       deleted:    abc
#
~
~
~
~
~
~
~
~
~
~
~
~
```

Now check the git log we can see revert “added abc “ in head

```
[waseemakram@waseem myproject.git % git revert 0d8cbef
 [akram 011bdc2] Revert "added abc"
  1 file changed, 0 insertions(+), 0 deletions(-)
  delete mode 100644 abc
[waseemakram@waseem myproject.git % git log --oneline
 011bdc2 (HEAD -> akram) Revert "added abc"
 0d8cbef added abc
 797e682 (origin/akram)  added file-a
 5f24119 Revert "added a file"
 814bb26 added a file
 68113b7 Revert " wrong commit"
```

10) Detached HEAD Challenge

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

—> check the log

```
waseemakram@waseem myproject.git % git log --oneline
011bdc2 (HEAD -> akram) Revert "added abc"
0d8cbef added abc
797e682 (origin/akram)  added file-a
5f24119 Revert "added a file"
814bb26 added a file
68113b7 Revert " wrong commit"
136f9c0  wrong commit
```

—>git checkout 011bdc2 (will Checkout the old commit (Detached HEAD))

```
waseemakram@waseem myproject.git % git checkout 011bdc2
Note: switching to '011bdc2'.

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 011bdc2 Revert "added abc"
```

—> Now Create a new branch from this commit

```
waseemakram@waseem myproject.git % git branch fix-abc
waseemakram@waseem myproject.git % git checkout fix-abc
Switched to branch 'fix-abc'
```

—>git branch

```
HEAD is now at 011bdc2 Revert "added abc"
waseemakram@waseem myproject.git % git branch
* (HEAD detached at 011bdc2)
  akram
  fix-abc
  main
  release
  waseem
waseemakram@waseem myproject.git %
```

We can see HEAD DETECTED AT 011bdc2

11) Git Hooks Challenge

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

—> go I side hooks directory
And create a file name pre-commit

```
  waseem
waseemakram@waseem myproject.git % cd .git/hooks

waseemakram@waseem hooks % vi pre-commit

waseemakram@waseem hooks % chmod +x pre-commit
```

Paste this rule for not config any commit with out jira-xxx

```
#!/bin/bash

# Get the commit message
MESSAGE=$(git log -1 --pretty=%B)

# Check if message starts with JIRA- followed by numbers
if ! grep -qE "^JIRA-[0-9]+" <<< "$MESSAGE"; then
    echo " Commit rejected!"
    echo "Your commit message MUST start with: JIRA-XXX"
    exit 1
fi

exit 0
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
```

Now give execute permission to file

```
  waseem
waseemakram@waseem myproject.git % cd .git/hooks

waseemakram@waseem hooks % vi pre-commit

waseemakram@waseem hooks % chmod +x pre-commit
```

We can see with out JIRA-XXX it is rejecting the commit

```
waseemakram@waseem myproject.git % git commit -m "update file"

 Commit rejected!
Your commit message MUST start with: JIRA-XXX
waseemakram@waseem myproject.git % touch bmw
waseemakram@waseem myproject.git % git add .
waseemakram@waseem myproject.git % git commit -m " added bmw"
 Commit rejected!
Your commit message MUST start with: JIRA-XXX
waseemakram@waseem myproject.git %
```

12) Squash Merge vs Rebase Merge

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

—>Created a two files and two commits for it in a new branch named feature-demo

```
akram          bmw          cat          dog
waseemakram@waseem myproject.git % git log --oneline
130746e (HEAD -> feature-demo)  added dog
73cf512 added cat
bda704d  add bmw
011bdc2 (fix-abc, akram) Revert "added abc"
0d8cbef added abc
797e682 (origin/akram)  added file-a
5f24119 Revert "added a file"
814bb26 added a file
68113b7 Revert " wrong commit"
```

git log —online —graph —decorate -n 5

```
waseemakram@waseem myproject.git % git log --oneline --graph --decorate -n 5

* 130746e (HEAD -> feature-demo)  added dog
* 73cf512 added cat
* bda704d  add bmw
* 011bdc2 (fix-abc, akram) Revert "added abc"
* 0d8cbef added abc
```

—>Git merge —squash feature-demo(Combines all commits into 1 new commit)

```
* 0a8cbef added abc
waseemakram@waseem myproject.git % git checkout main

Switched to branch 'main'
Your branch is up to date with 'origin/main'.
waseemakram@waseem myproject.git % ls
akram           file1           file2           README.md       was             world
waseemakram@waseem myproject.git % git merge --squash feature-demo

Updating fe6aee2..130746e
Fast-forward
Squash commit -- not updating HEAD
 bmw    | 0
 cat    | 0
 dog    | 0
 file-a | 1 +
 file-b | 1 +
 5 files changed, 2 insertions(+)
 create mode 100644 bmw
 create mode 100644 cat
 create mode 100644 dog
 create mode 100644 file-a
 create mode 100644 file-b
```

Only **one** commit appears on main (`squashed feature-demo`)
Create one combined commit
—> git commit -m “ squashed feature-demo “

```
create mode 100644 file-b
waseemakram@waseem myproject.git % git commit -m "squashed feature-demo"

[main e6db90c] squashed feature-demo
 5 files changed, 2 insertions(+)
 create mode 100644 bmw
 create mode 100644 cat
 create mode 100644 dog
 create mode 100644 file-a
 create mode 100644 file-b
waseemakram@waseem myproject.git % git log --oneline -3

e6db90c (HEAD -> main) squashed feature-demo
fe6aee2 (origin/main, origin/HEAD)  added file in main
1fa5df9 added modified file in main
waseemakram@waseem myproject.git % git reset --hard HEAD~1
```

Now check git log
We can see squashed feature-demo

It has combined all commits in to one “squashed feature-demo”

Rebase

Git reset —hard HEAD~1
Will make second commit the HEAD now

```
waseemakram@waseem myproject.git % git log --oneline -3

e6db90c (HEAD -> main) squashed feature-demo
fe6aee2 (origin/main, origin/HEAD)  added file in main
1fa5df9 added modified file in main
waseemakram@waseem myproject.git % git reset --hard HEAD~1

HEAD is now at fe6aee2  added file in main
waseemakram@waseem myproject.git % git checkout feature-demo

Switched to branch 'feature-demo'
```

Git rebase main(Rebase feature onto current `main`)

```
Switched to branch 'feature-demo'
waseemakram@waseem myproject.git % git rebase main

Current branch feature-demo is up to date.
waseemakram@waseem myproject.git % git checkout main
git merge feature-demo

Switched to branch 'main'
Your branch is up to date with 'origin/main'.
Updating fe6aee2..130746e
Fast-forward
 bmw    | 0
 cat    | 0
 dog    | 0
 file-a | 1 +
 file-b | 1 +
 5 files changed, 2 insertions(+)
 create mode 100644 bmw
 create mode 100644 cat
 create mode 100644 dog
 create mode 100644 file-a
 create mode 100644 file-b
```

Now
—>git checkout main (. Switch to main)
—> git merge feature-demo (this will merge the two branches)

We can see in log the commits are one by one

```
waseemakram@waseem myproject.git % git log --oneline -5

130746e (HEAD -> main, feature-demo)  added dog
73cf512 added cat
bda704d  add bmw
011bdc2 (fix-abc, akram) Revert "added abc"
0d8cbef added abc
waseemakram@waseem myproject.git %
```

Squash merge:

- Combines all commits into **one commit**
- History becomes clean
- Looks like a single change

Rebase merge:

- Replays commits one-by-one
- Produces **all original commits** on main
- History is linear and detailed

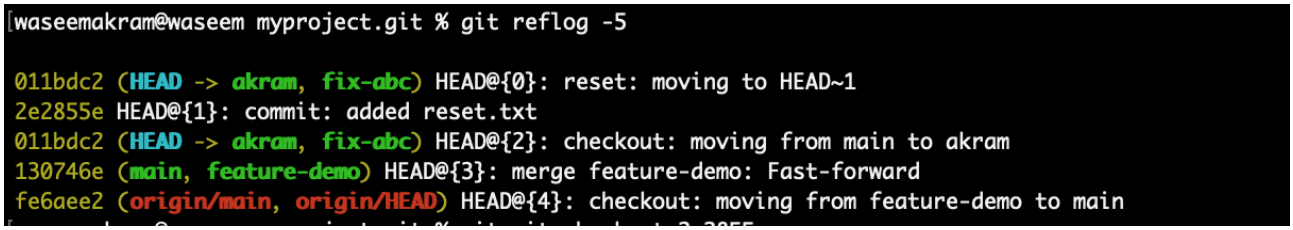
14) Recover Lost Commit

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

—> we created a file and
—> git add .
—> git commit -m

We checked its log
And used reset hard
—> git reset —hard HEAD~1
This remove the head commit
Now to recovering process
—> lets check reflog

—> git reflog will show us the information record



—> for recovering it we copied the commit of deleted commit

—> git checkout 2e2855e this will recover the commit

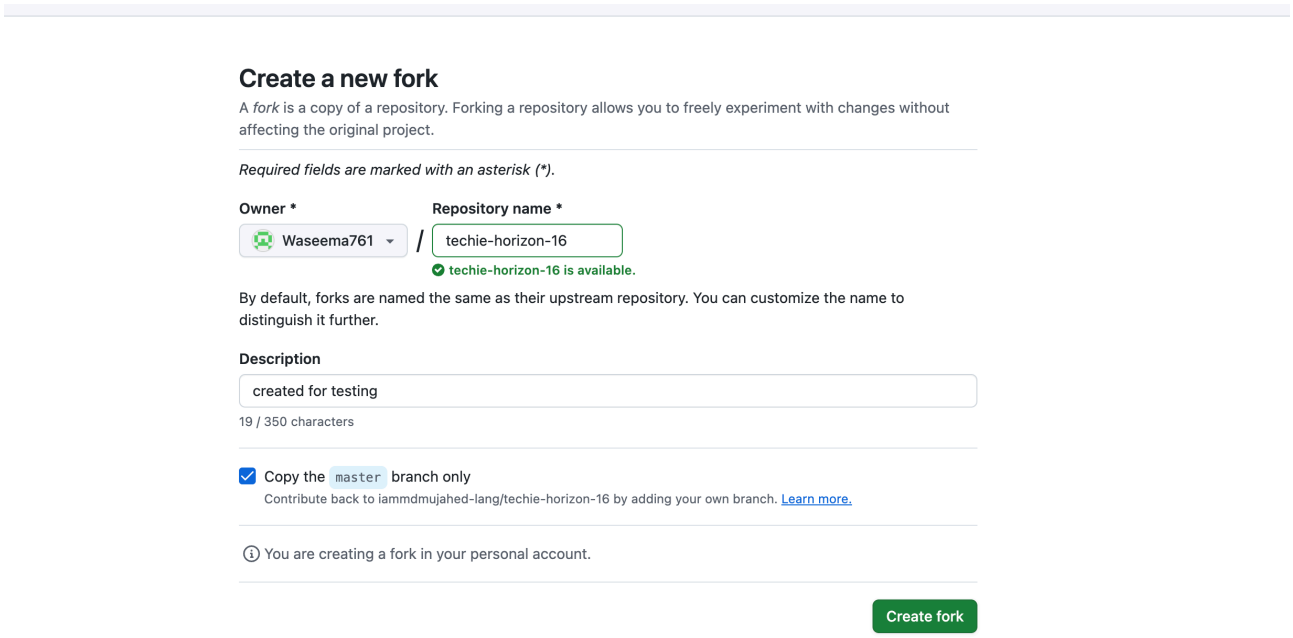
```
waseemakram@waseem myproject.git % git checkout 2e2855e
Note: switching to '2e2855e'.

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 2e2855e  added reset.txt
waseemakram@waseem myproject.git % git log --oneline -6
2e2855e (HEAD)  added reset.txt
011bdc2 (fix-abc, akram) Revert "added abc"
0d8cbef added abc
797e682 (origin/akram)  added file-a
5f24119 Revert "added a file"
814bb26 added a file
waseemakram@waseem myproject.git %
```

We can conform it by git log
We can see commit 2e2855e is recovered to head

13) Fork & Pull Request Workflow

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

—> forked the repo from other user named techie-horizon-16



Now after forking the repo copy the url and past it
—> git clone < ssh url of that repo >
—> cd techie-horizon-16 (now we are inside clone of that fork repo of other user)

```
akram          bmw            cat            dog            file-a         file-b         file1          file2          READ
[waseemakram@waseem myproject.git % git clone git@github.com:Waseema761/techie-horizon-16.git
Cloning into 'techie-horizon-16'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 3 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (3/3), done.
[waseemakram@waseem myproject.git % ls
akram                  cat                    file-a                 file1                  README.md              was
bmw                    dog                    file-b                 file2                  techie-horizon-16      world
[waseemakram@waseem myproject.git % cd techie-horizon-16
```

Now modify any file from that repo
And
—> git add .
—> git commit -m
—> git push

```
[waseemakram@waseem myproject.git % cd techie-horizon-16
[waseemakram@waseem techie-horizon-16 % ls
 file
[waseemakram@waseem techie-horizon-16 % vi file
[waseemakram@waseem techie-horizon-16 % git status
 On branch master
 Your branch is up to date with 'origin/master'.

 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
```

```
[waseemakram@waseem techie-horizon-16 % git add .
[waseemakram@waseem techie-horizon-16 % git commit -m " modified file "
 [master 656d950]  modified file
  1 file changed, 1 insertion(+)
[waseemakram@waseem techie-horizon-16 % git status
 On branch master
 Your branch is ahead of 'origin/master' by 1 commit.
```

Now git push to push the branch changes

```
[waseemakram@waseem techie-horizon-16 % git push
 Enumerating objects: 5, done.
 Counting objects: 100% (5/5), done.
 Writing objects: 100% (3/3), 253 bytes | 253.00 KiB/s, done.
 Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
 To github.com:Waseema761/techie-horizon-16.git
    7bd8d83..656d950  master -> master
 waseemakram@waseem techie-horizon-16 %
```

We
can see file is modified and pushed