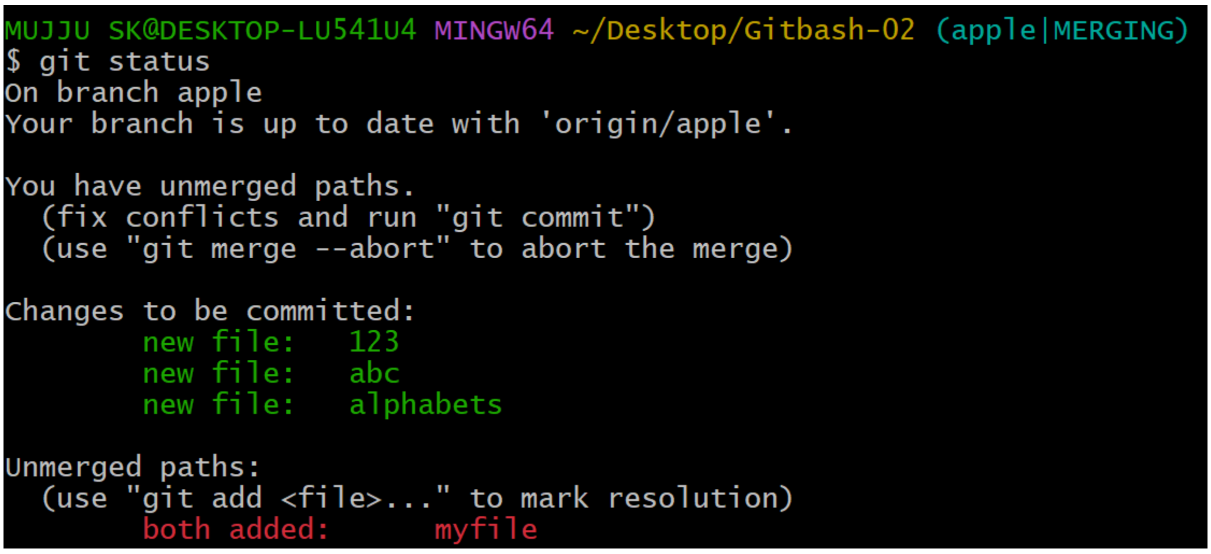
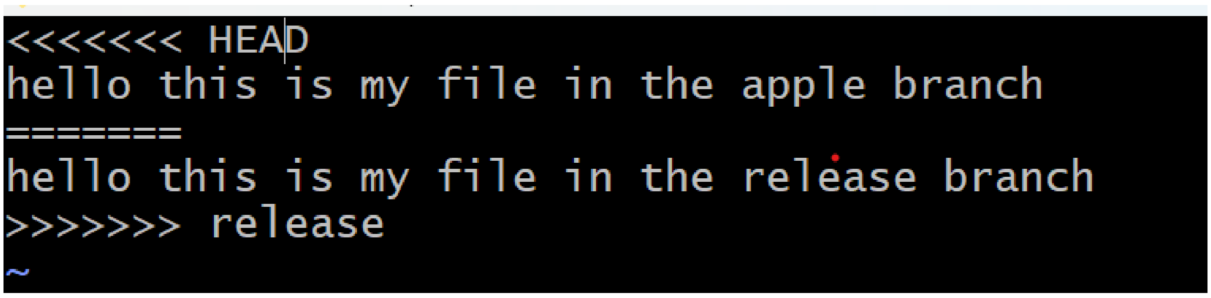
1. **Resolve Merge Conflicts**
   - **Create a merge conflict intentionally (two users editing the same line).**
   - **Resolve the conflict and push the changes.**

Create two files in different branches with same file names and try to merge two branches then it show merge conflict

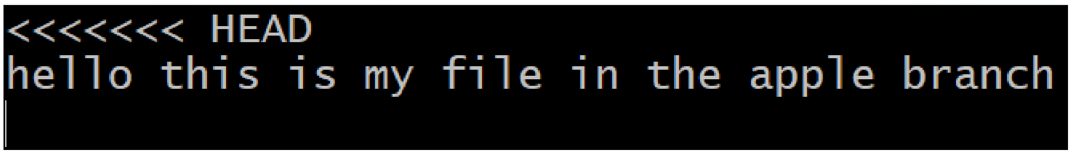


```
MUJJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop/Gitbash-02 (apple|MERGING)
$ git status
On branch apple
Your branch is up to date with 'origin/apple'.

You have unmerged paths.
  (fix conflicts and run "git commit")
  (use "git merge --abort" to abort the merge)

Changes to be committed:
        new file:   123
        new file:   abc
        new file:   alphabets

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

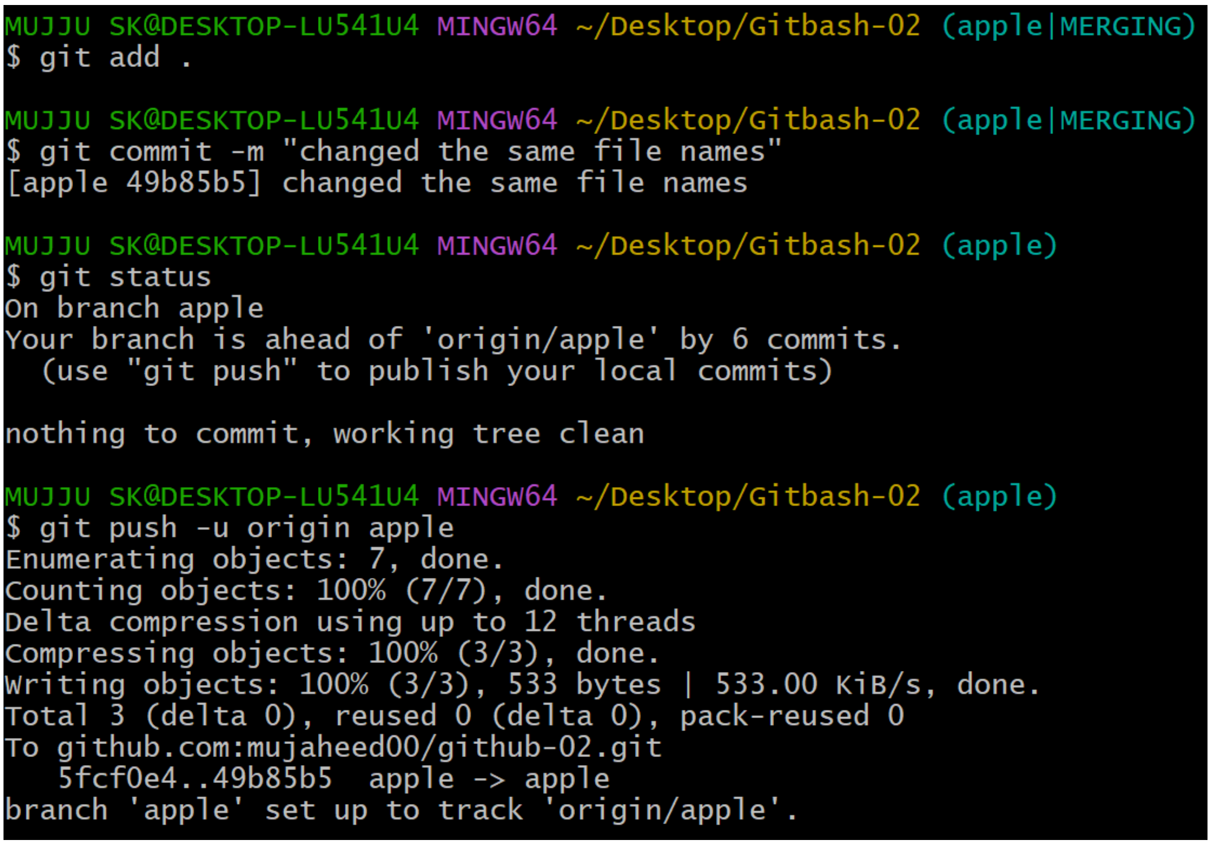


```
<<<<<<< HEAD
hello this is my file in the apple branch
=======
hello this is my file in the release branch
>>>>>>> release
~
```

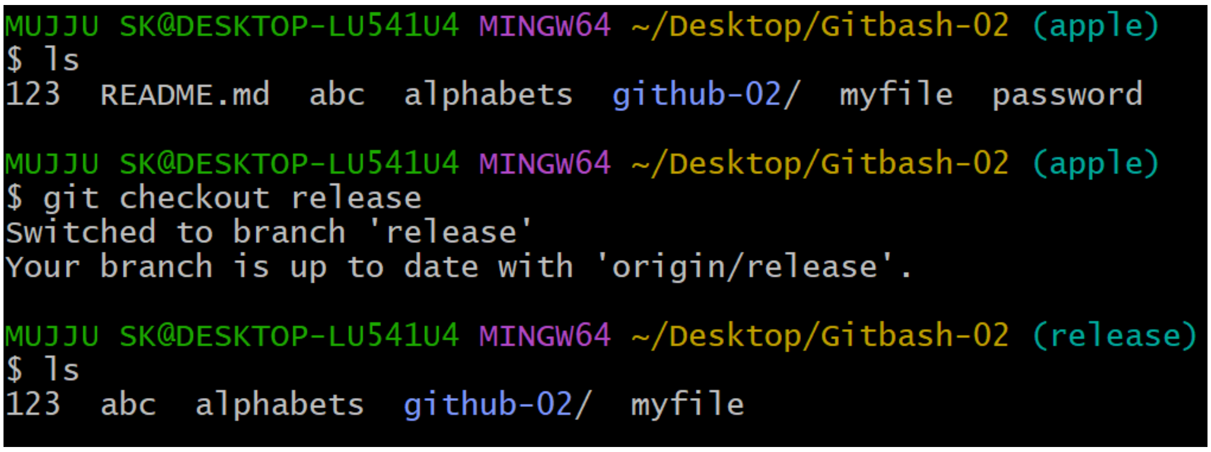
Edit the file with



```
<<<<<<< HEAD
hello this is my file in the apple branch
```



```
MUJJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop/Gitbash-02 (apple|MERGING)
$ git add .

MUJJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop/Gitbash-02 (apple|MERGING)
$ git commit -m "changed the same file names"
[apple 49b85b5] changed the same file names

MUJJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop/Gitbash-02 (apple)
$ git status
On branch apple
Your branch is ahead of 'origin/apple' by 6 commits.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean

MUJJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop/Gitbash-02 (apple)
$ git push -u origin apple
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 12 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 533 bytes | 533.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:mujaheed00/github-02.git
   5fcf0e4..49b85b5  apple -> apple
branch 'apple' set up to track 'origin/apple'.
```



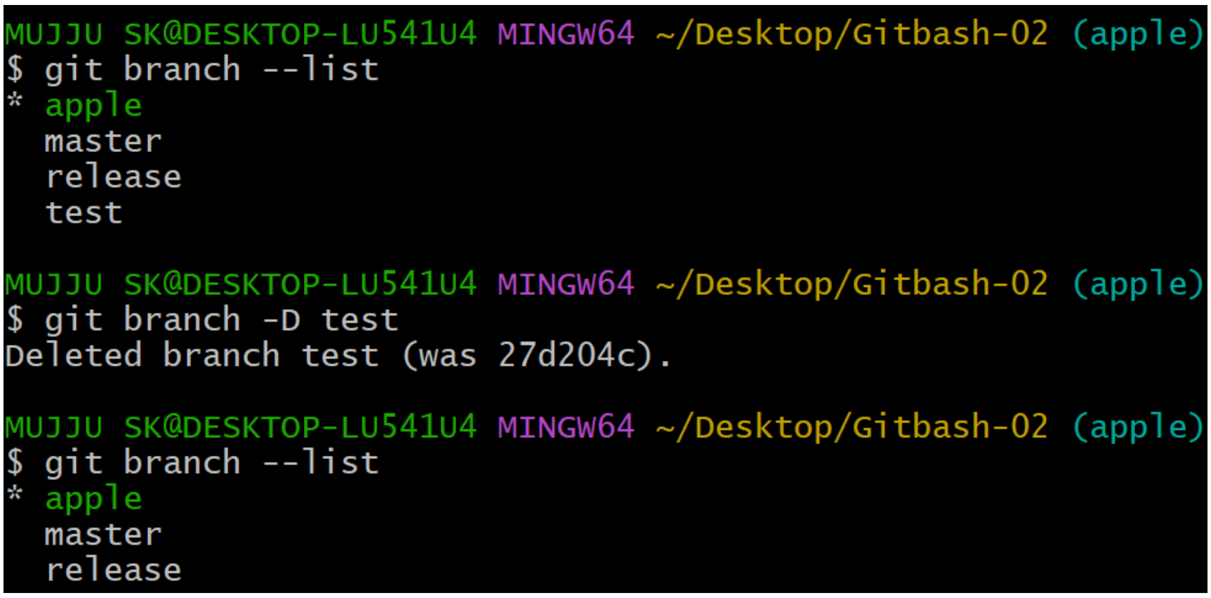
```
MUJJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop/Gitbash-02 (apple)
$ ls
123  README.md  abc  alphabets  github-02/  myfile  password

MUJJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop/Gitbash-02 (apple)
$ git checkout release
Switched to branch 'release'
Your branch is up to date with 'origin/release'.

MUJJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop/Gitbash-02 (release)
$ ls
123  abc  alphabets  github-02/  myfile
```

2. Recover Deleted Branch

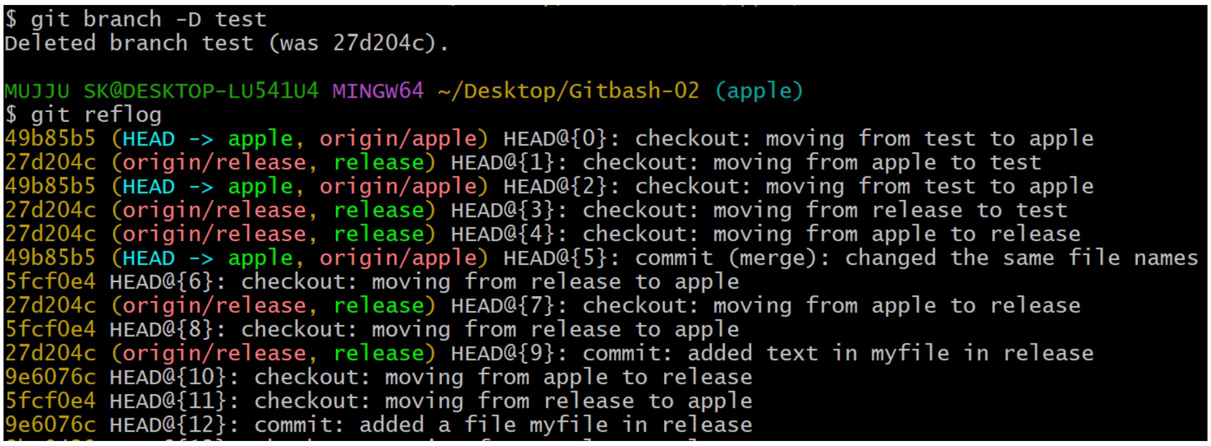
- Delete a local branch and then recover it using the reflog.

Create a branch named as test and delete it by using git branch -D test.

Then the branch will be deleted.

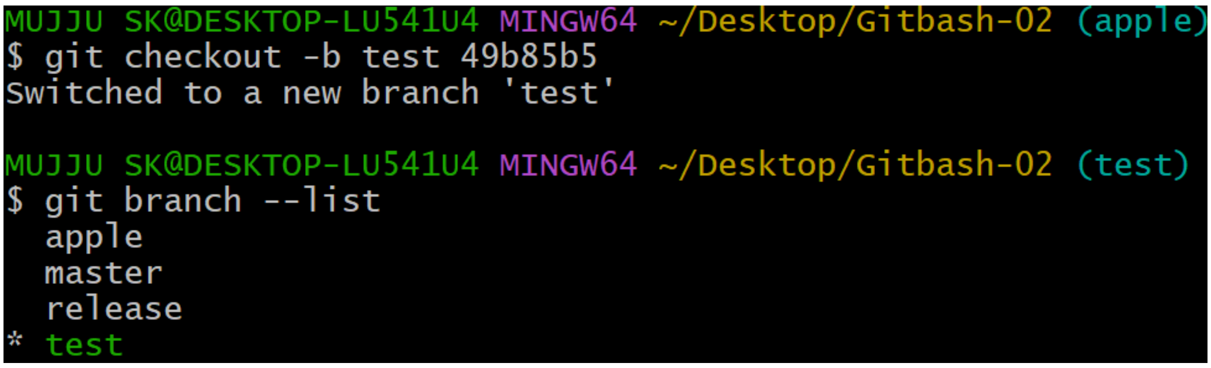


```
MUJJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop/Gitbash-02 (apple)
$ git branch --list
* apple
  master
  release
  test

MUJJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop/Gitbash-02 (apple)
$ git branch -D test
Deleted branch test (was 27d204c).

MUJJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop/Gitbash-02 (apple)
$ git branch --list
* apple
  master
  release
```



```
$ git branch -D test
Deleted branch test (was 27d204c).

MUJJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop/Gitbash-02 (apple)
$ git reflog
49b85b5 (HEAD -> apple, origin/apple) HEAD@{0}: checkout: moving from test to apple
27d204c (origin/release, release) HEAD@{1}: checkout: moving from apple to test
49b85b5 (HEAD -> apple, origin/apple) HEAD@{2}: checkout: moving from test to apple
27d204c (origin/release, release) HEAD@{3}: checkout: moving from release to test
27d204c (origin/release, release) HEAD@{4}: checkout: moving from apple to release
49b85b5 (HEAD -> apple, origin/apple) HEAD@{5}: commit (merge): changed the same file names
5fcf0e4 HEAD@{6}: checkout: moving from release to apple
27d204c (origin/release, release) HEAD@{7}: checkout: moving from apple to release
5fcf0e4 HEAD@{8}: checkout: moving from release to apple
27d204c (origin/release, release) HEAD@{9}: commit: added text in myfile in release
9e6076c HEAD@{10}: checkout: moving from apple to release
5fcf0e4 HEAD@{11}: checkout: moving from release to apple
9e6076c HEAD@{12}: commit: added a file myfile in release
```

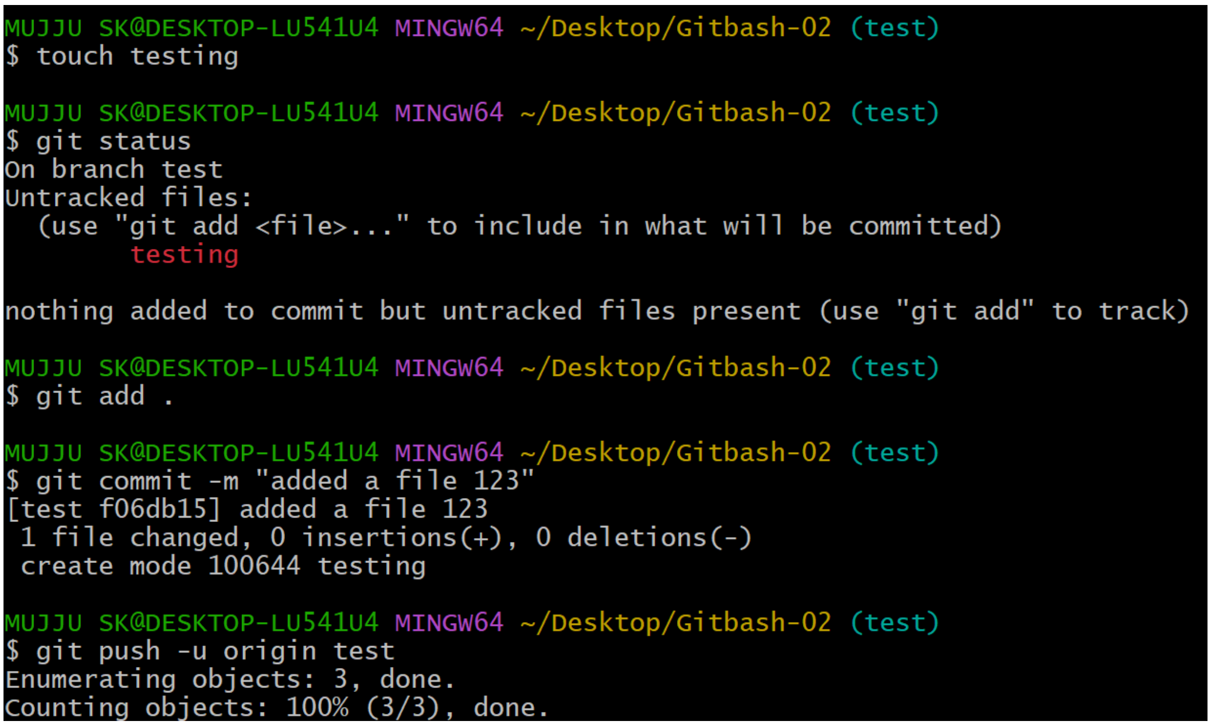
By using the command git reflog we will get the commit id of deleted file at HEAD@(0) . By using git checkout -b branchname commit_id we get our deleted branch.



```
MUJJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop/Gitbash-02 (apple)
$ git checkout -b test 49b85b5
Switched to a new branch 'test'

MUJJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop/Gitbash-02 (test)
$ git branch --list
  apple
  master
  release
* test
```

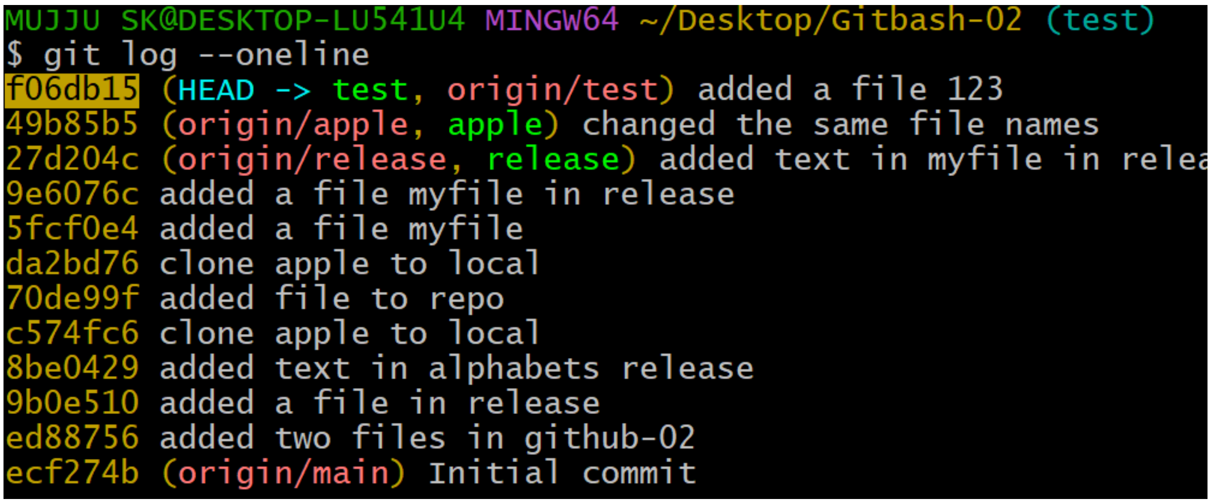
3. Undo Wrong Push

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



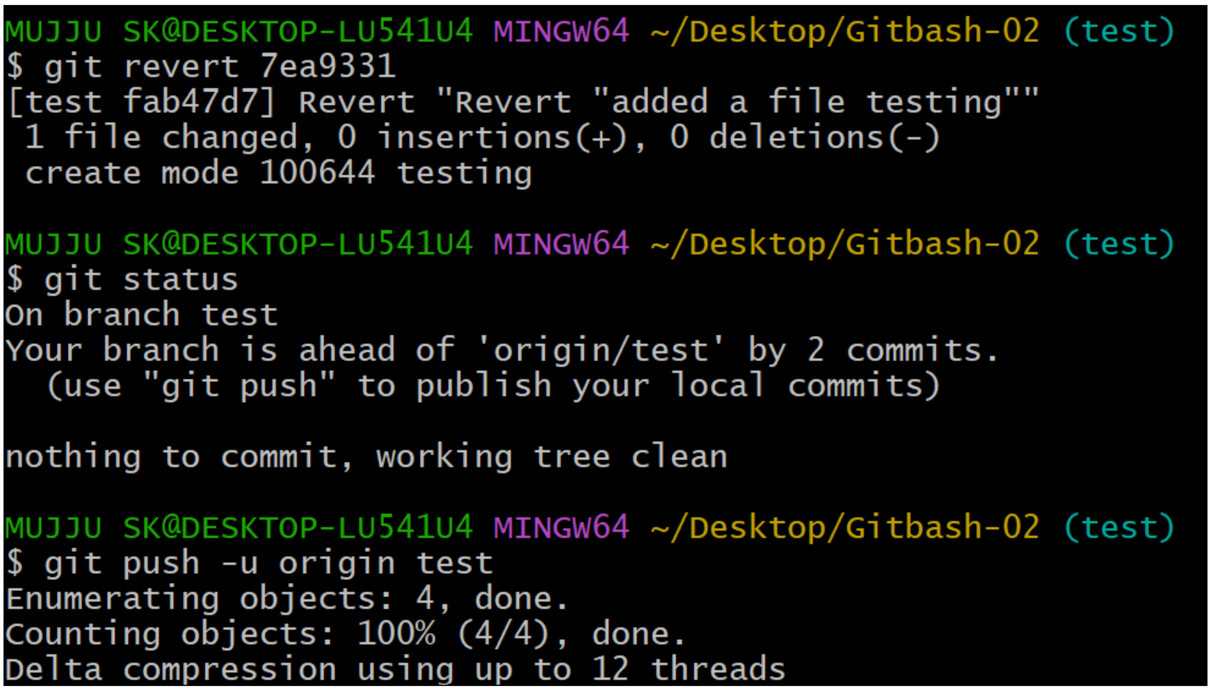
```
MUJJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop/Gitbash-02 (test)
$ touch testing

MUJJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop/Gitbash-02 (test)
$ git status
On branch test
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        testing

nothing added to commit but untracked files present (use "git add" to track)

MUJJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop/Gitbash-02 (test)
$ git add .

MUJJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop/Gitbash-02 (test)
$ git commit -m "added a file 123"
[test f06db15] added a file 123
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 testing

MUJJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop/Gitbash-02 (test)
$ git push -u origin test
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
```

Created a file named as testing and commit that with wrong file name.



```
MUJJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop/Gitbash-02 (test)
$ git log --oneline
f06db15 (HEAD -> test, origin/test) added a file 123
49b85b5 (origin/apple, apple) changed the same file names
27d204c (origin/release, release) added text in myfile in relea
9e6076c added a file myfile in release
5fcf0e4 added a file myfile
da2bd76 clone apple to local
70de99f added file to repo
c574fc6 clone apple to local
8be0429 added text in alphabets release
9b0e510 added a file in release
ed88756 added two files in github-02
ecf274b (origin/main) Initial commit
```

By using git commit id use git revert commit id we can edit the commit message

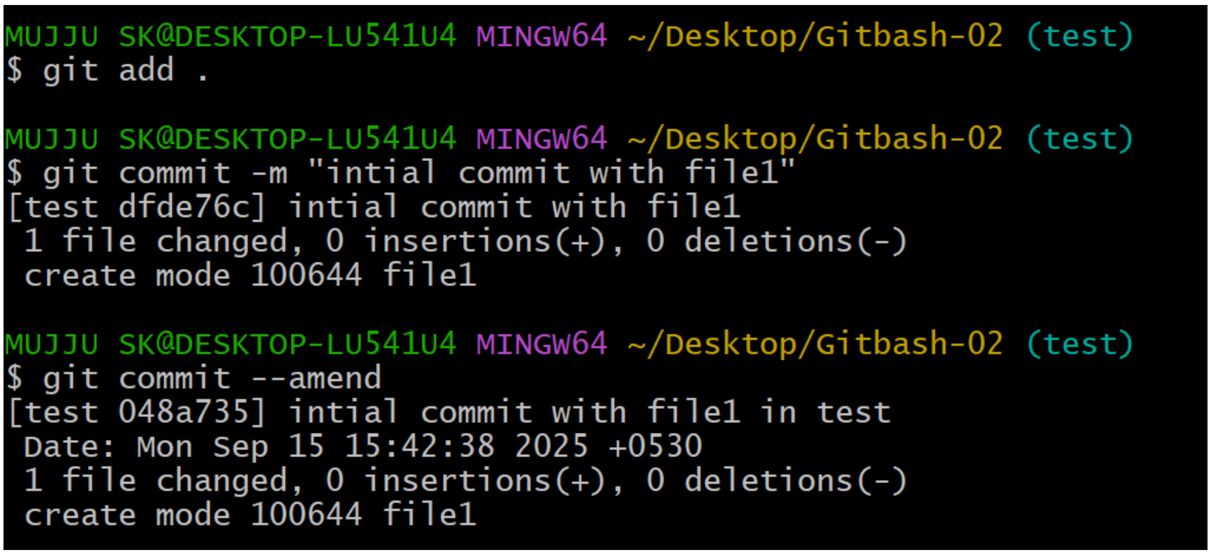


```
MUJJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop/Gitbash-02 (test)
$ git revert 7ea9331
[test fab47d7] Revert "Revert "added a file testing""
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 testing

MUJJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop/Gitbash-02 (test)
$ git status
On branch test
Your branch is ahead of 'origin/test' by 2 commits.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean

MUJJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop/Gitbash-02 (test)
$ git push -u origin test
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 12 threads
```

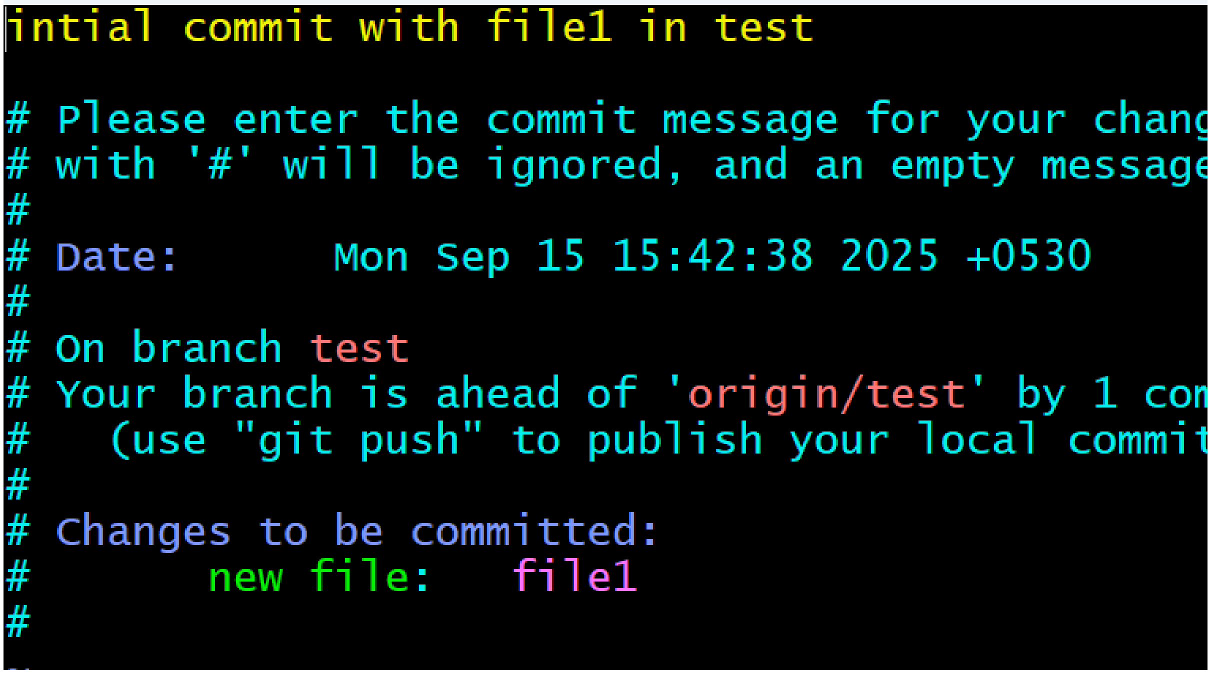
4. Amend a Commit

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

Create a file named as file1 and added to git add . and committed file1 and I need to change the commit message then use git commit –amend.



```
MUJJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop/Gitbash-02 (test)
$ git add .

MUJJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop/Gitbash-02 (test)
$ git commit -m "intial commit with file1"
[test dfde76c] intial commit with file1
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 file1

MUJJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop/Gitbash-02 (test)
$ git commit --amend
[test 048a735] intial commit with file1 in test
 Date: Mon Sep 15 15:42:38 2025 +0530
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 file1
```

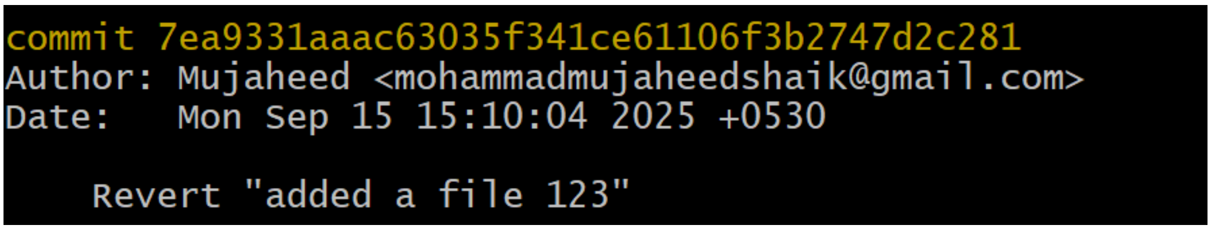


```
intial commit with file1 in test

# Please enter the commit message for your chang
# with '#' will be ignored, and an empty message
#
# Date:      Mon Sep 15 15:42:38 2025 +0530
#
# On branch test
# Your branch is ahead of 'origin/test' by 1 com
#   (use "git push" to publish your local commit
#
# Changes to be committed:
#       new file:   file1
#
```

5. Cherry-pick a Commit

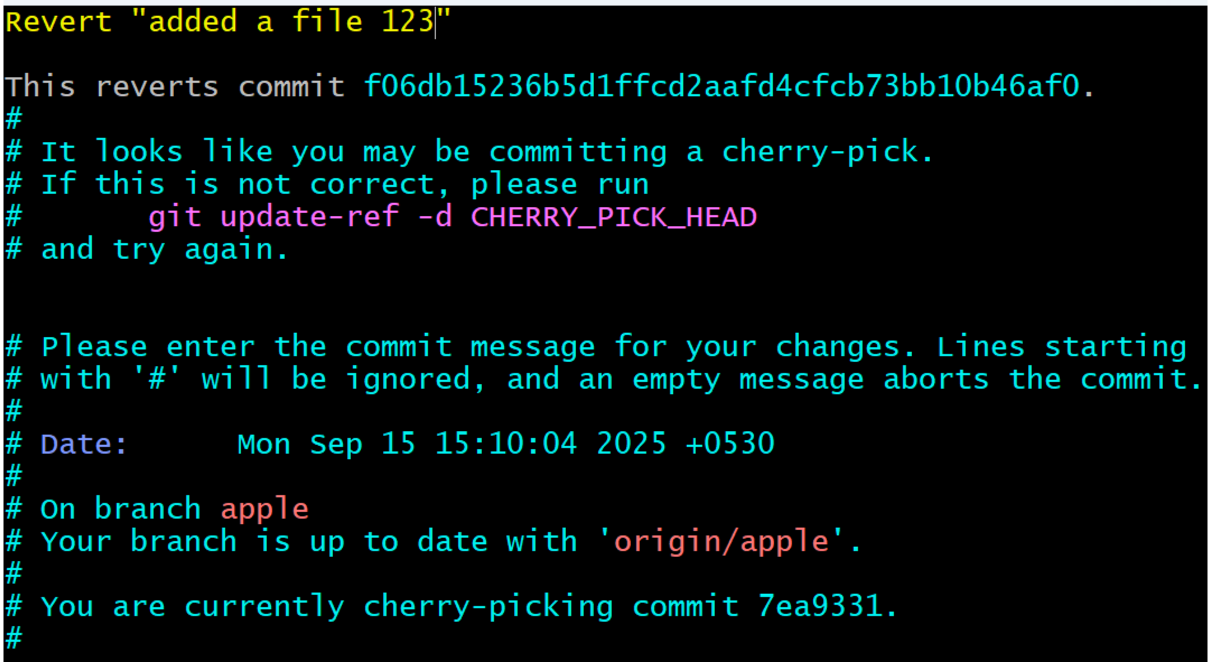
- Take a specific commit from one branch and apply it to another branch.

Take a commitid from the test branch.

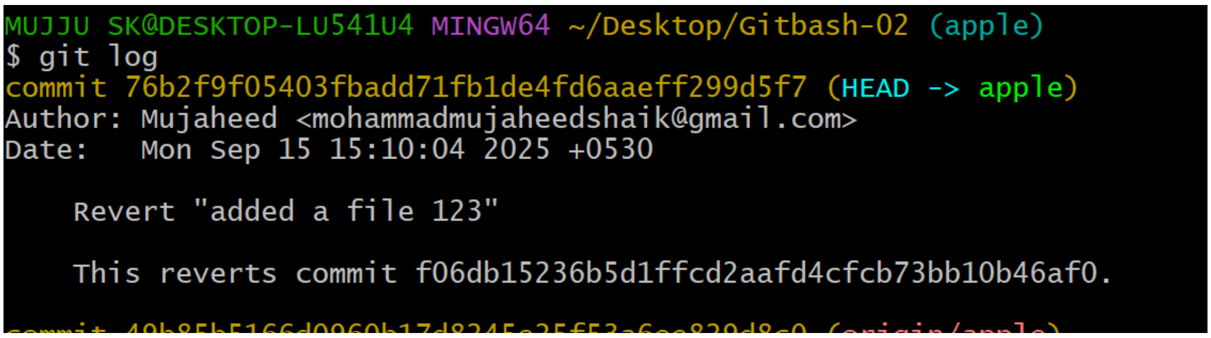


```
commit 7ea9331aaac63035f341ce61106f3b2747d2c281
Author: Mujaheed <mohammadmujaheedshaik@gmail.com>
Date:   Mon Sep 15 15:10:04 2025 +0530

    Revert "added a file 123"
```

By using the command git cherry-pick commit id from apple branch then the editor will open. Save the changes as you need then the commit id will come to the another branch.



```
Revert "added a file 123"

This reverts commit f06db15236b5d1ffcd2aafd4cfcb73bb10b46af0.
#
# It looks like you may be committing a cherry-pick.
# If this is not correct, please run
#       git update-ref -d CHERRY_PICK_HEAD
# and try again.


# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
#
# Date:      Mon Sep 15 15:10:04 2025 +0530
#
# On branch apple
# Your branch is up to date with 'origin/apple'.
#
# You are currently cherry-picking commit 7ea9331.
#
```



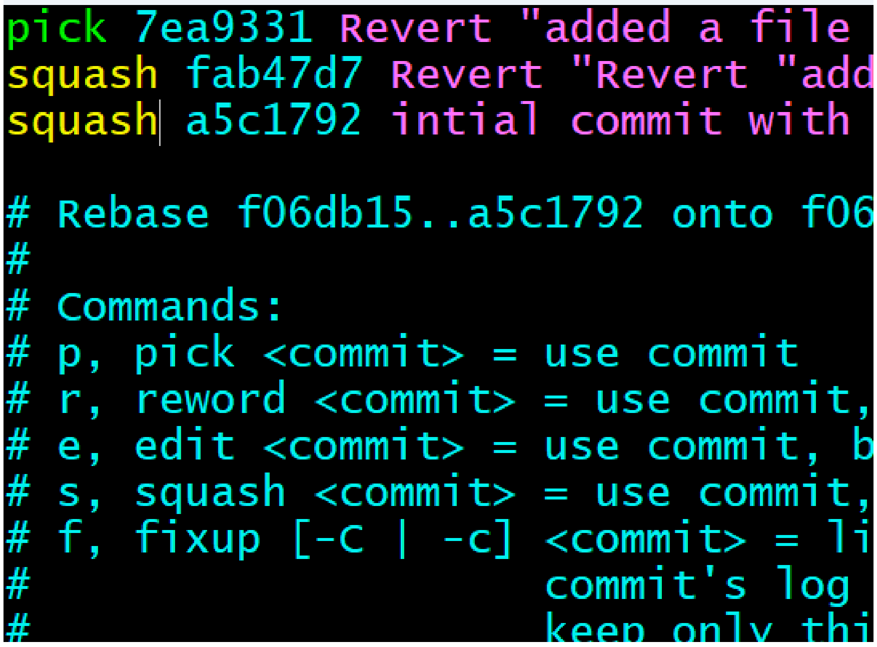
```
MUJJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop/Gitbash-02 (apple)
$ git log
commit 76b2f9f05403fbadd71fb1de4fd6aaeff299d5f7 (HEAD -> apple)
Author: Mujaheed <mohammadmujaheedshaik@gmail.com>
Date:   Mon Sep 15 15:10:04 2025 +0530

    Revert "added a file 123"

    This reverts commit f06db15236b5d1ffcd2aafd4cfcb73bb10b46af0.
```

6. Interactive Rebase

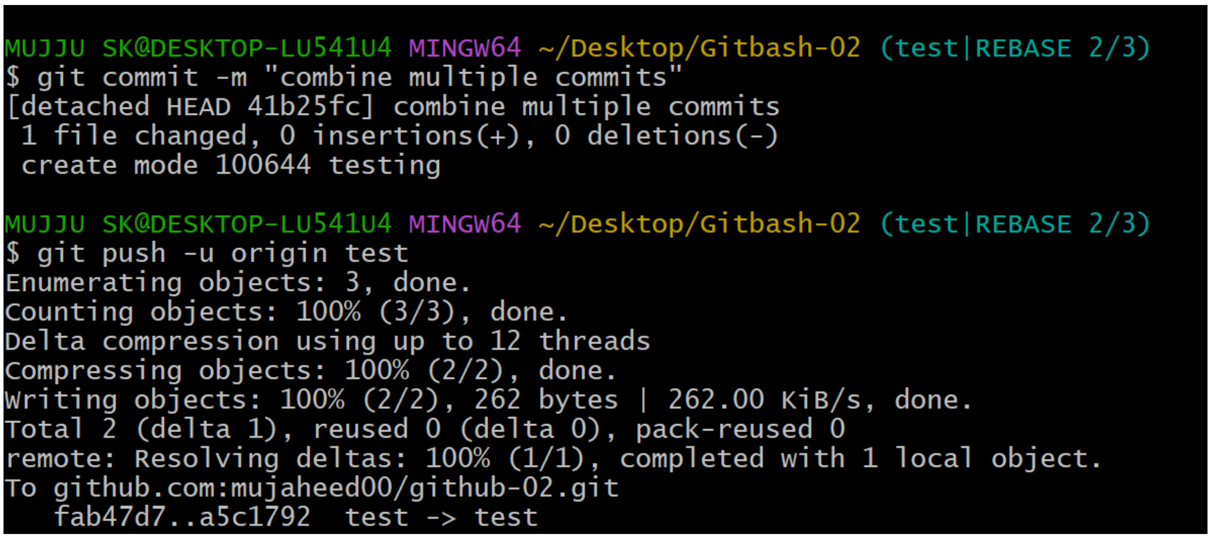
- **Reorder and squash multiple commits into a single clean commit.**

Use command git rebase -I HEAD~N here N is the number of commits you need to combine .

In next step the editor will open keep one commit as pick and remaining all as squash, then it will combine the multiple commits to single commit.



```
pick 7ea9331 Revert "added a file
squash fab47d7 Revert "Revert "add
squash a5c1792 intial commit with

# Rebase f06db15..a5c1792 onto f06
#
# Commands:
# p, pick <commit> = use commit
# r, reword <commit> = use commit,
# e, edit <commit> = use commit, b
# s, squash <commit> = use commit,
# f, fixup [-C | -c] <commit> = li
#                    commit's log
#                    keep only thi
```

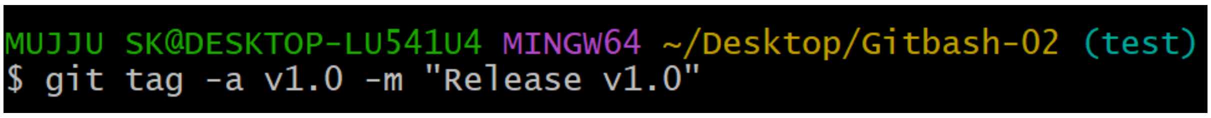


```
MUJJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop/Gitbash-02 (test|REBASE 2/3)
$ git commit -m "combine multiple commits"
[detached HEAD 41b25fc] combine multiple commits
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 testing

MUJJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop/Gitbash-02 (test|REBASE 2/3)
$ git push -u origin test
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 12 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (2/2), 262 bytes | 262.00 KiB/s, done.
Total 2 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To github.com:mujaheed00/github-02.git
   fab47d7..a5c1792  test -> test
```

7. Tagging & Release

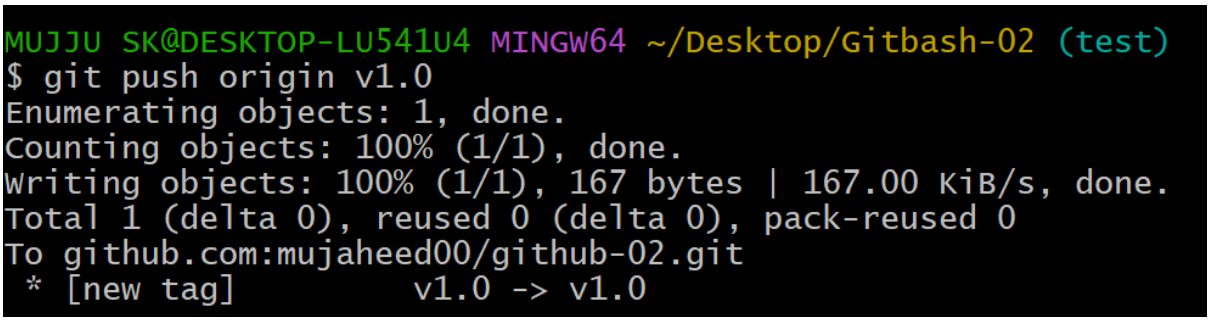
- **Create a version tag (v1.0), push it to GitHub, then delete and restore it.**

Create a tag of version v1 by using the command :git tag -a v1.0 -m “Release v1.0”



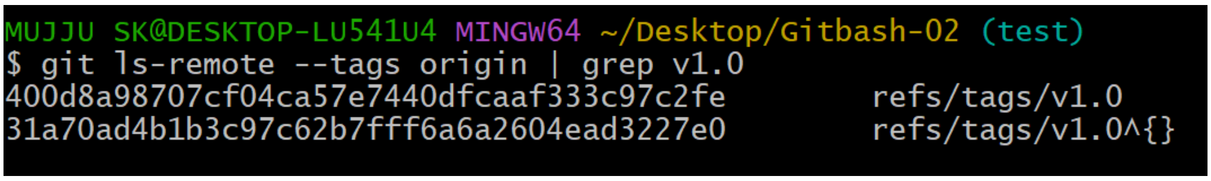
```
MUJJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop/Gitbash-02 (test)
$ git tag -a v1.0 -m "Release v1.0"
```

Push the tag to to GitHub by using git push origin v1.0.



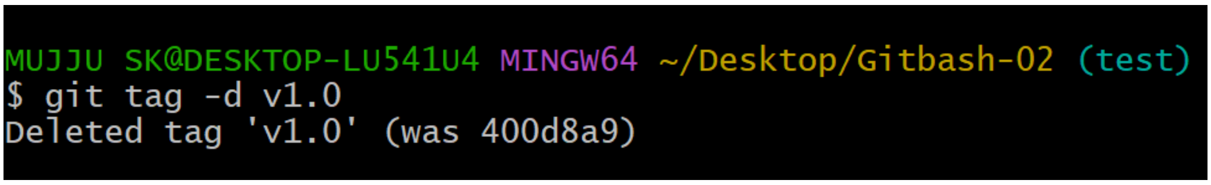
```
MUJJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop/Gitbash-02 (test)
$ git push origin v1.0
Enumerating objects: 1, done.
Counting objects: 100% (1/1), done.
Writing objects: 100% (1/1), 167 bytes | 167.00 KiB/s, done.
Total 1 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:mujaheed00/github-02.git
 * [new tag]         v1.0 -> v1.0
```

To verify the tag we need to use the command : git ls-remote --tags origin | grep v1.0.



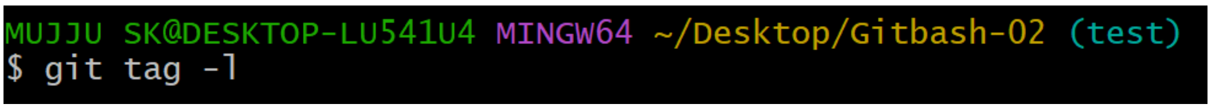
```
MUJJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop/Gitbash-02 (test)
$ git ls-remote --tags origin | grep v1.0
400d8a98707cf04ca57e7440dfcaaf333c97c2fe        refs/tags/v1.0
31a70ad4b1b3c97c62b7fff6a6a2604ead3227e0        refs/tags/v1.0^{}
```

If we want to delete the tag we will use the command :git tag -d v1.0



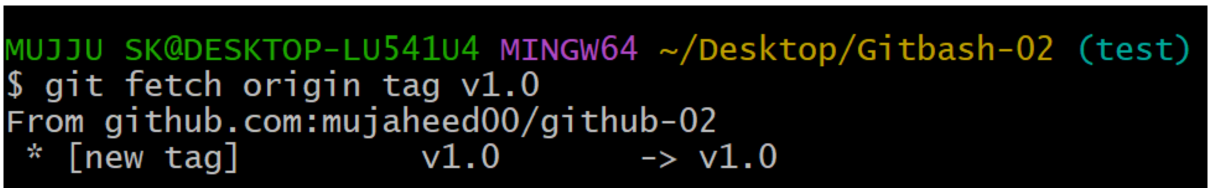
```
MUJJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop/Gitbash-02 (test)
$ git tag -d v1.0
Deleted tag 'v1.0' (was 400d8a9)
```

To check the list of tags we need to use



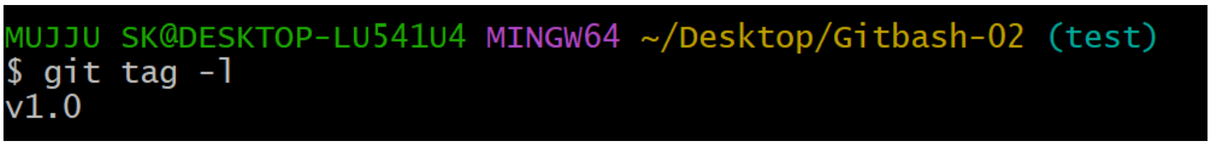
```
MUJJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop/Gitbash-02 (test)
$ git tag -l
```

To restore the tag we need to use the command :git fetch origin tag v1.0.



```
MUJJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop/Gitbash-02 (test)
$ git fetch origin tag v1.0
From github.com:mujaheed00/github-02
 * [new tag]         v1.0       -> v1.0
```

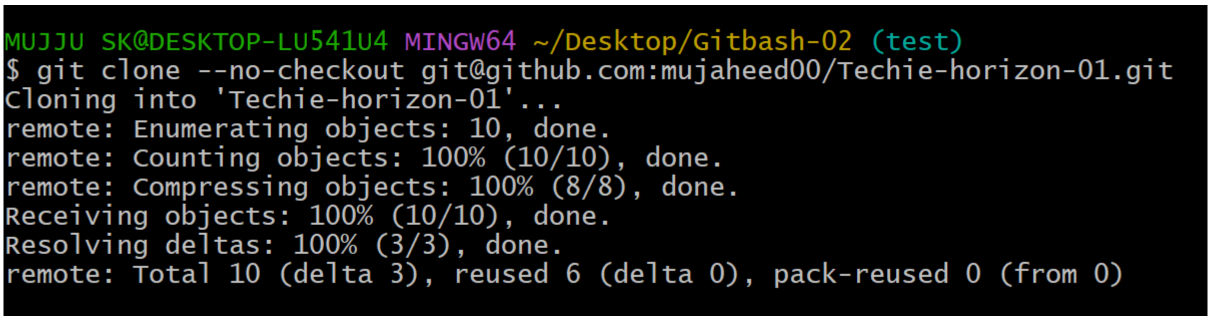
If want to check the tag list :git tag -l.


```
MUJJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop/Gitbash-02 (test)
$ git tag -l
v1.0
```

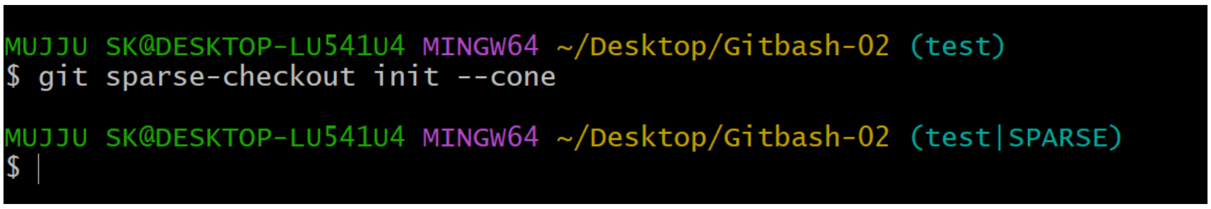
8. Clone with Sparse Checkout

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

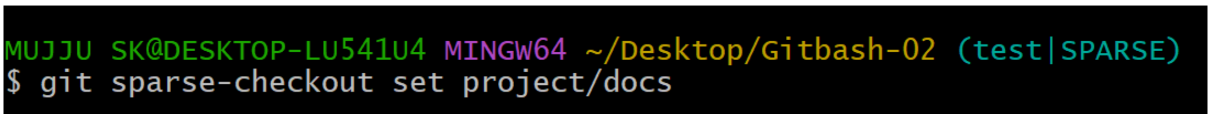
Use git clone --no-checkout and repolink.


```
MUJJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop/Gitbash-02 (test)
$ git clone --no-checkout git@github.com:mujaheed00/Techie-horizon-01.git
Cloning into 'Techie-horizon-01'...
remote: Enumerating objects: 10, done.
remote: Counting objects: 100% (10/10), done.
remote: Compressing objects: 100% (8/8), done.
Receiving objects: 100% (10/10), done.
Resolving deltas: 100% (3/3), done.
remote: Total 10 (delta 3), reused 6 (delta 0), pack-reused 0 (from 0)
```

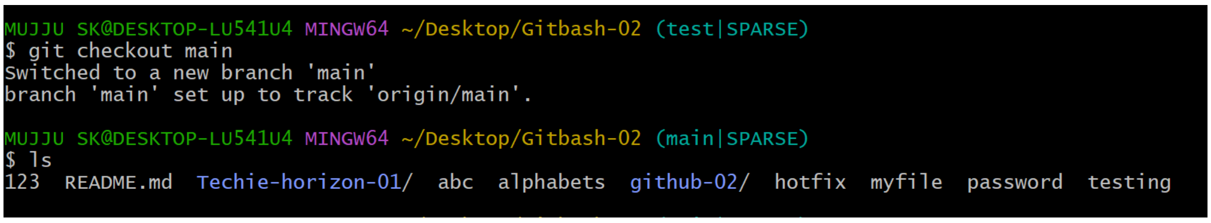
Use git sparse-checkout init --cone.


```
MUJJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop/Gitbash-02 (test)
$ git sparse-checkout init --cone

MUJJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop/Gitbash-02 (test|SPARSE)
$ |
```

If you want the folder project/doc use :git sparse-checkout set projects/docs.


```
MUJJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop/Gitbash-02 (test|SPARSE)
$ git sparse-checkout set project/docs
```

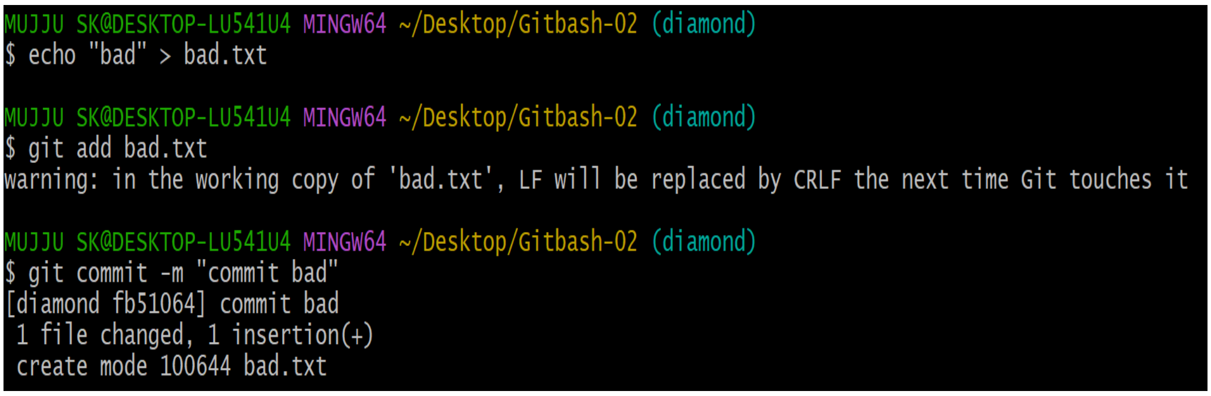
Use git checkout main to clone with sparse checkout.


```
MUJJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop/Gitbash-02 (test|SPARSE)
$ git checkout main
Switched to a new branch 'main'
branch 'main' set up to track 'origin/main'.

MUJJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop/Gitbash-02 (main|SPARSE)
$ ls
123  README.md  Techie-horizon-01/  abc  alphabets  github-02/  hotfix  myfile  password  testing
```

9. Reset vs Revert Challenge

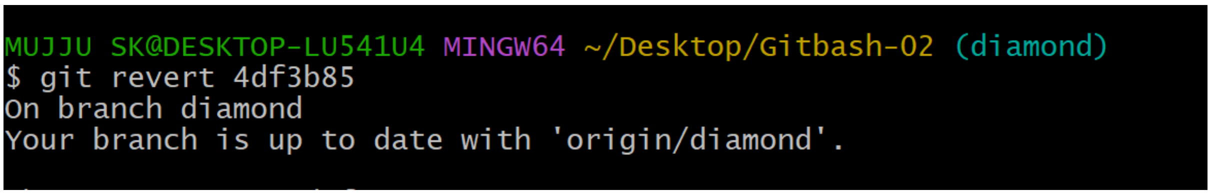
- **Demonstrate the difference between git reset --hard and git revert in a repo.**

Create a file bad and commit it.



```
MUJJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop/Gitbash-02 (diamond)
$ echo "bad" > bad.txt

MUJJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop/Gitbash-02 (diamond)
$ git add bad.txt
warning: in the working copy of 'bad.txt', LF will be replaced by CRLF the next time Git touches it

MUJJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop/Gitbash-02 (diamond)
$ git commit -m "commit bad"
[diamond fb51064] commit bad
 1 file changed, 1 insertion(+)
 create mode 100644 bad.txt
```

By using command git revert commitid we can revert from central repo.



```
MUJJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop/Gitbash-02 (diamond)
$ git revert 4df3b85
On branch diamond
Your branch is up to date with 'origin/diamond'.
```

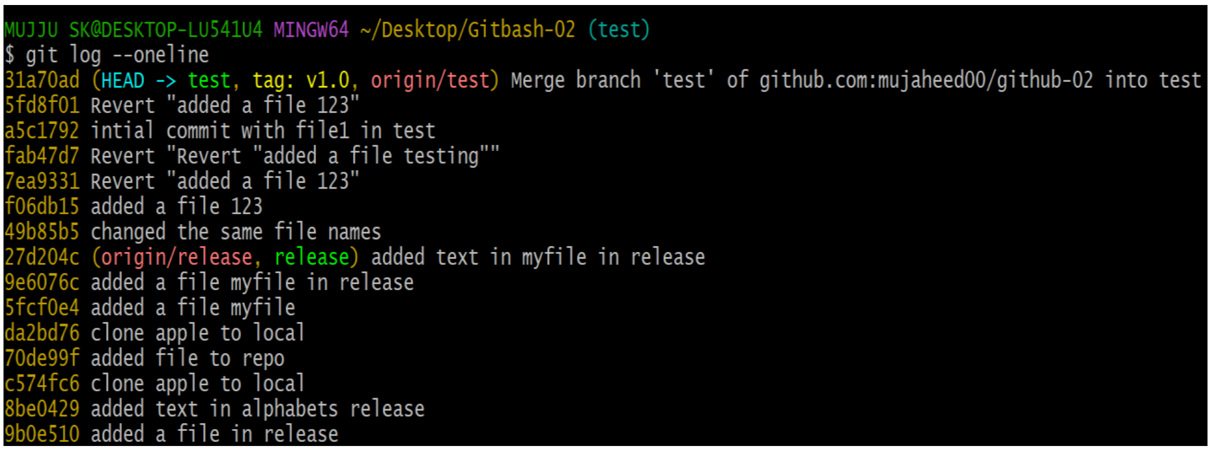
Use reset → when working locally and you want to erase/rewrite history
before pushing.

Use revert → when you already pushed the commit to GitHub (shared repo).

10. Detached HEAD Challenge

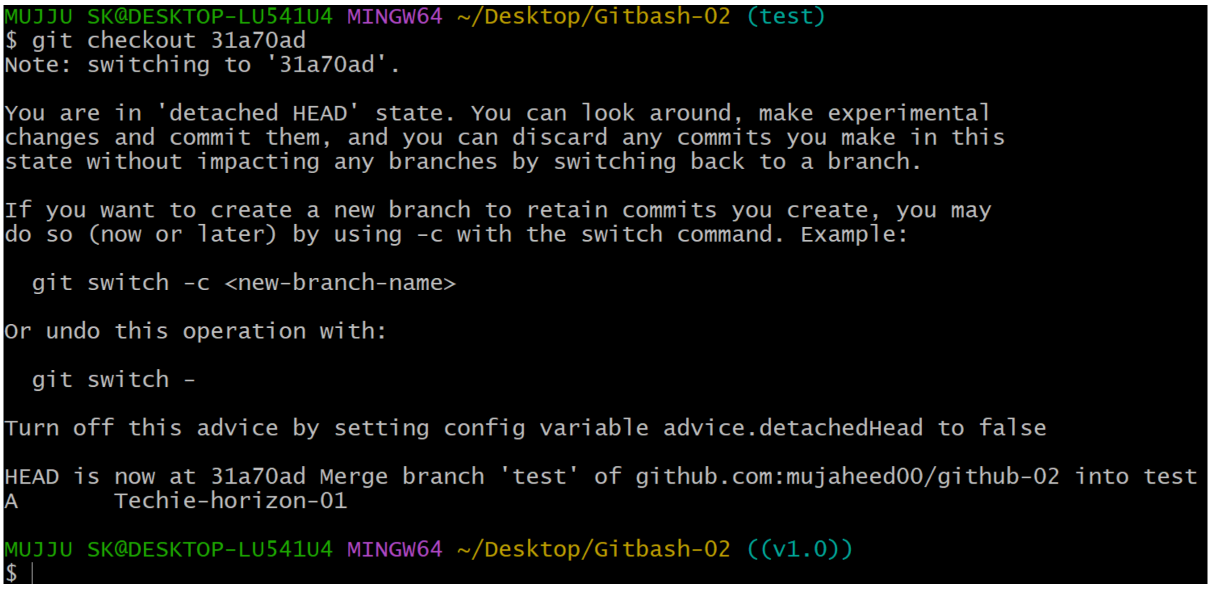
- **Checkout a specific commit (detached HEAD state) and create a new branch from it.**

Use the command git log --oneline to check the commits



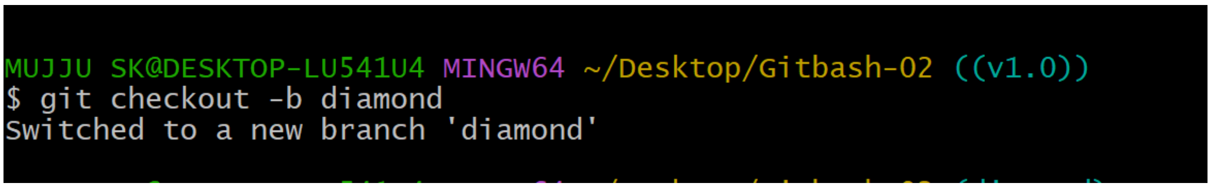
```
MUJJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop/Gitbash-02 (test)
$ git log --oneline
31a70ad (HEAD -> test, tag: v1.0, origin/test) Merge branch 'test' of github.com:mujaheed00/github-02 into test
5fd8f01 Revert "added a file 123"
a5c1792 intial commit with file1 in test
fab47d7 Revert "Revert "added a file testing""
7ea9331 Revert "added a file 123"
f06db15 added a file 123
49b85b5 changed the same file names
27d204c (origin/release, release) added text in myfile in release
9e6076c added a file myfile in release
5fcf0e4 added a file myfile
da2bd76 clone apple to local
70de99f added file to repo
c574fc6 clone apple to local
8be0429 added text in alphabets release
9b0e510 added a file in release
```

Select the HEAD commit id and execute a command git checkout commitid then it will goes to the commitid.



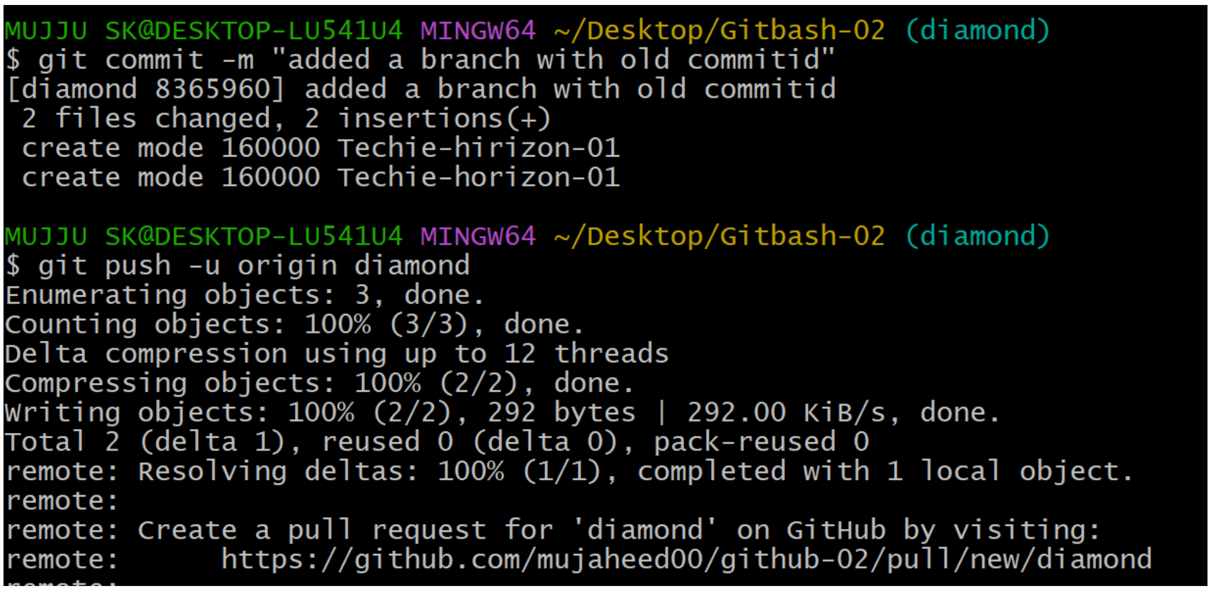
```
MUJJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop/Gitbash-02 (test)
$ git checkout 31a70ad
Note: switching to '31a70ad'.

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 31a70ad Merge branch 'test' of github.com:mujaheed00/github-02 into test
A       Techie-horizon-01

MUJJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop/Gitbash-02 ((v1.0))
$ 
```

Add a branch to that commit by using git checkout -b newbranchname.



```
MUJJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop/Gitbash-02 ((v1.0))
$ git checkout -b diamond
Switched to a new branch 'diamond'
```

Push the file into repository



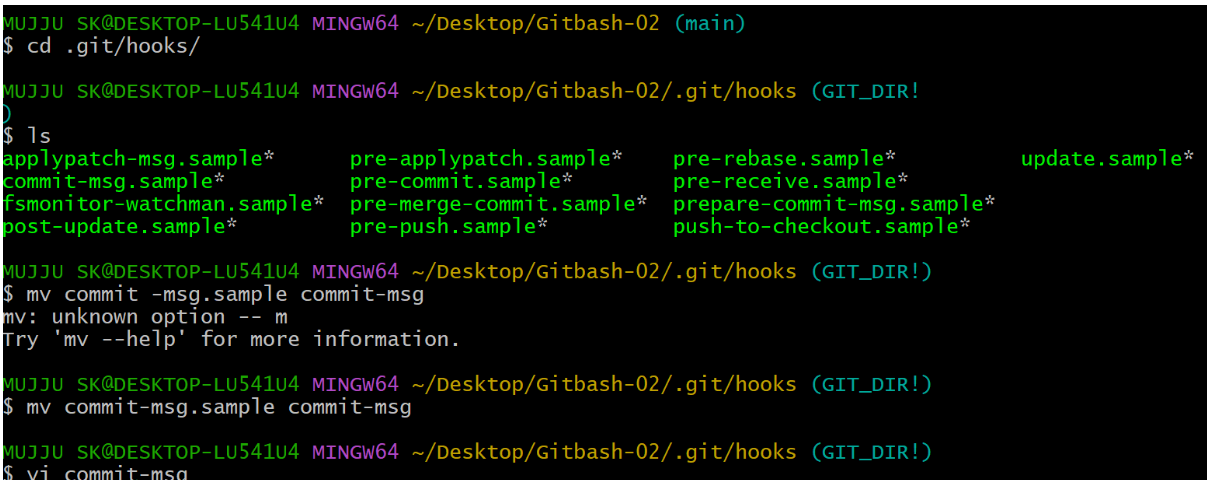
```
MUJJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop/Gitbash-02 (diamond)
$ git commit -m "added a branch with old commitid"
[diamond 8365960] added a branch with old commitid
 2 files changed, 2 insertions(+)
 create mode 160000 Techie-hirizon-01
 create mode 160000 Techie-horizon-01

MUJJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop/Gitbash-02 (diamond)
$ git push -u origin diamond
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 12 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (2/2), 292 bytes | 292.00 KiB/s, done.
Total 2 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
remote:
remote: Create a pull request for 'diamond' on GitHub by visiting:
remote:      https://github.com/mujaheed00/github-02/pull/new/diamond
```

11. Git Hooks Challenge

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

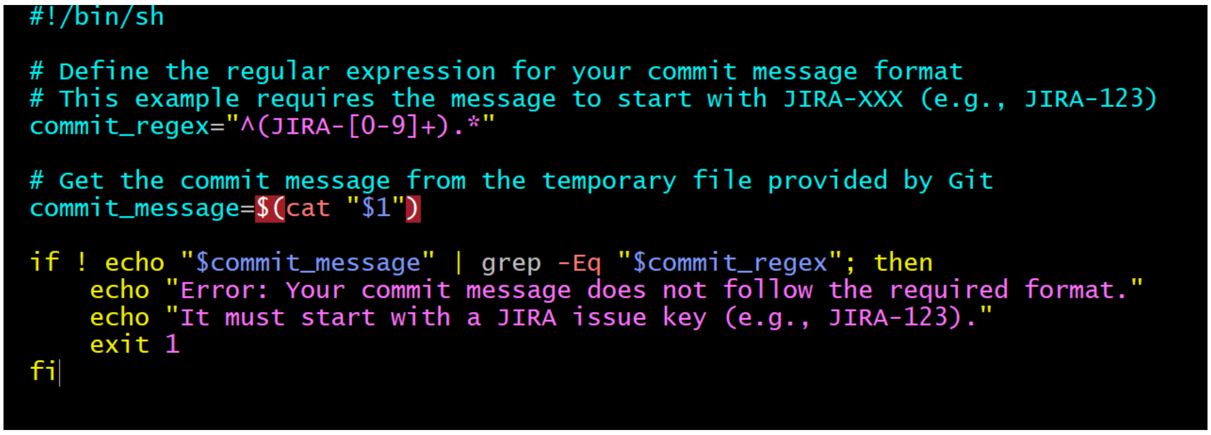
Change the directory to .git/hooks/.

Mv commit -msg.sample commit-msg.



```
MUJJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop/Gitbash-02 (main)
$ cd .git/hooks/

MUJJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop/Gitbash-02/.git/hooks (GIT_DIR!
)
$ ls
applypatch-msg.sample*      pre-applypatch.sample*    pre-rebase.sample*          update.sample*
commit-msg.sample*          pre-commit.sample*        pre-receive.sample*
fsmonitor-watchman.sample*  pre-merge-commit.sample*  prepare-commit-msg.sample*
post-update.sample*         pre-push.sample*          push-to-checkout.sample*

MUJJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop/Gitbash-02/.git/hooks (GIT_DIR!)
$ mv commit -msg.sample commit-msg
mv: unknown option -- m
Try 'mv --help' for more information.

MUJJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop/Gitbash-02/.git/hooks (GIT_DIR!)
$ mv commit-msg.sample commit-msg

MUJJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop/Gitbash-02/.git/hooks (GIT_DIR!)
$ vi commit-msg
```

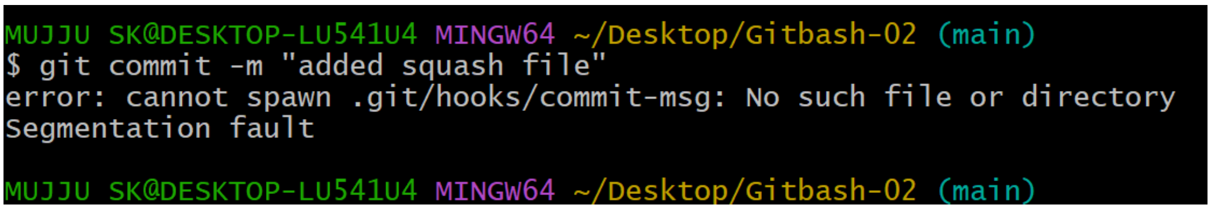
Edit the file with this script



```
#!/bin/sh

# Define the regular expression for your commit message format
# This example requires the message to start with JIRA-XXX (e.g., JIRA-123)
commit_regex="^(JIRA-[0-9]+).*"

# Get the commit message from the temporary file provided by Git
commit_message=$(cat "$1")

if ! echo "$commit_message" | grep -Eq "$commit_regex"; then
    echo "Error: Your commit message does not follow the required format."
    echo "It must start with a JIRA issue key (e.g., JIRA-123)."
    exit 1
fi
```

Give the permissions to change mode to chmod +x commit-msg.and commit the msg .

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



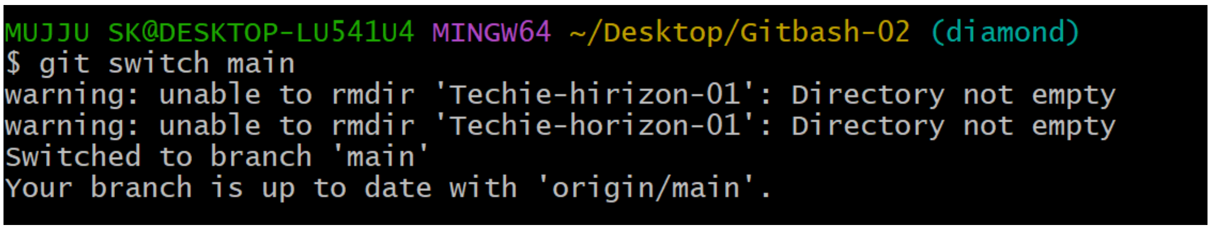
```
MUJJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop/Gitbash-02 (main)
$ git commit -m "added squash file"
error: cannot spawn .git/hooks/commit-msg: No such file or directory
Segmentation fault

MUJJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop/Gitbash-02 (main)
```

12. Squash Merge vs Rebase Merge

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

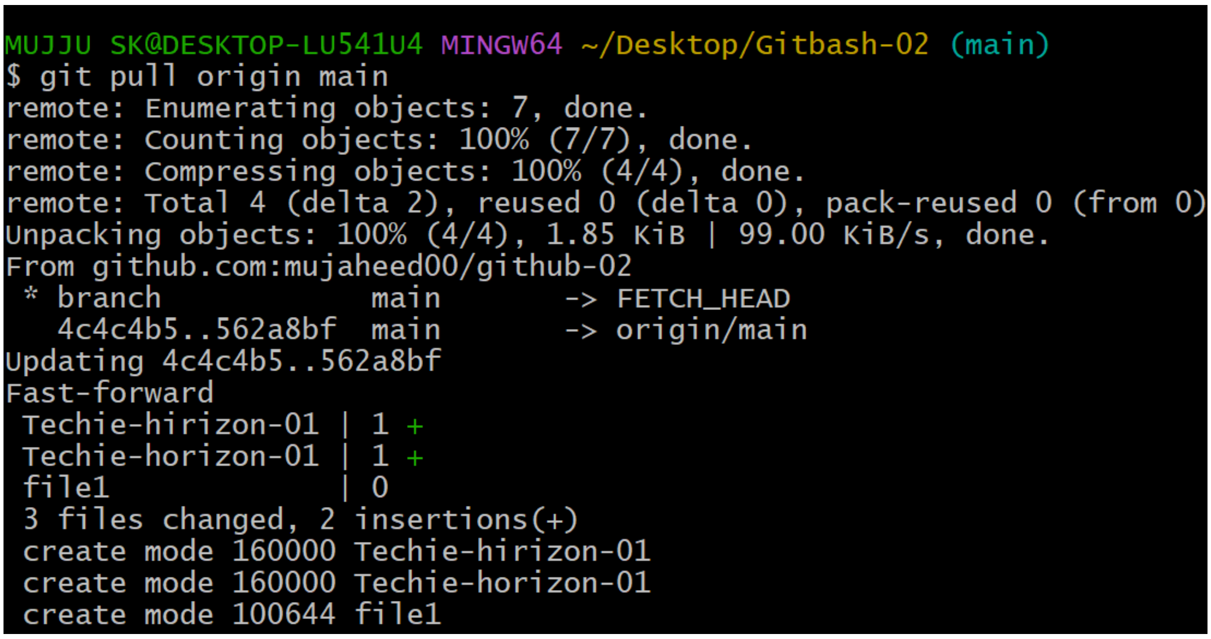
Squash merge:

Switch to main branch



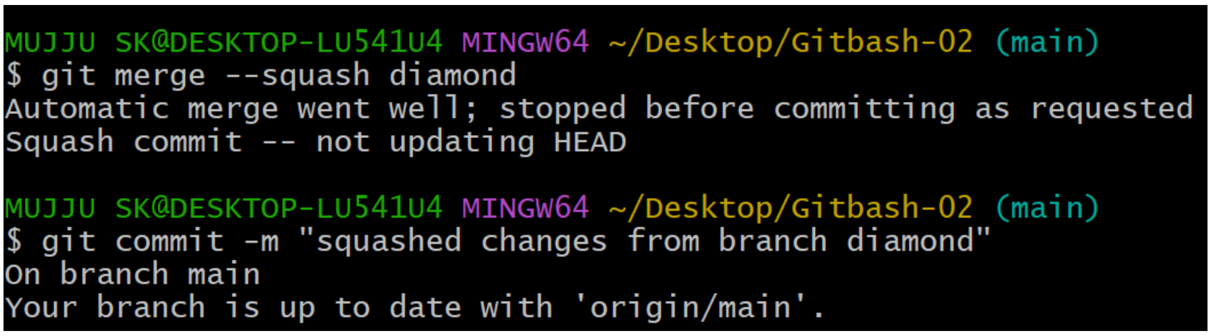
```
MUJJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop/Gitbash-02 (diamond)
$ git switch main
warning: unable to rmdir 'Techie-hirizon-01': Directory not empty
warning: unable to rmdir 'Techie-horizon-01': Directory not empty
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
```

Use git pull origin main



```
MUJJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop/Gitbash-02 (main)
$ git pull origin main
remote: Enumerating objects: 7, done.
remote: Counting objects: 100% (7/7), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 4 (delta 2), reused 0 (delta 0), pack-reused 0 (from 0)
Unpacking objects: 100% (4/4), 1.85 KiB | 99.00 KiB/s, done.
From github.com:mujaheed00/github-02
 * branch            main       -> FETCH_HEAD
   4c4c4b5..562a8bf  main       -> origin/main
Updating 4c4c4b5..562a8bf
Fast-forward
 Techie-hirizon-01 | 1 +
 Techie-horizon-01 | 1 +
 file1             | 0
 3 files changed, 2 insertions(+)
 create mode 160000 Techie-hirizon-01
 create mode 160000 Techie-horizon-01
 create mode 100644 file1
```

Use the command git merge --squash diamond and commit the changes and push into repository.

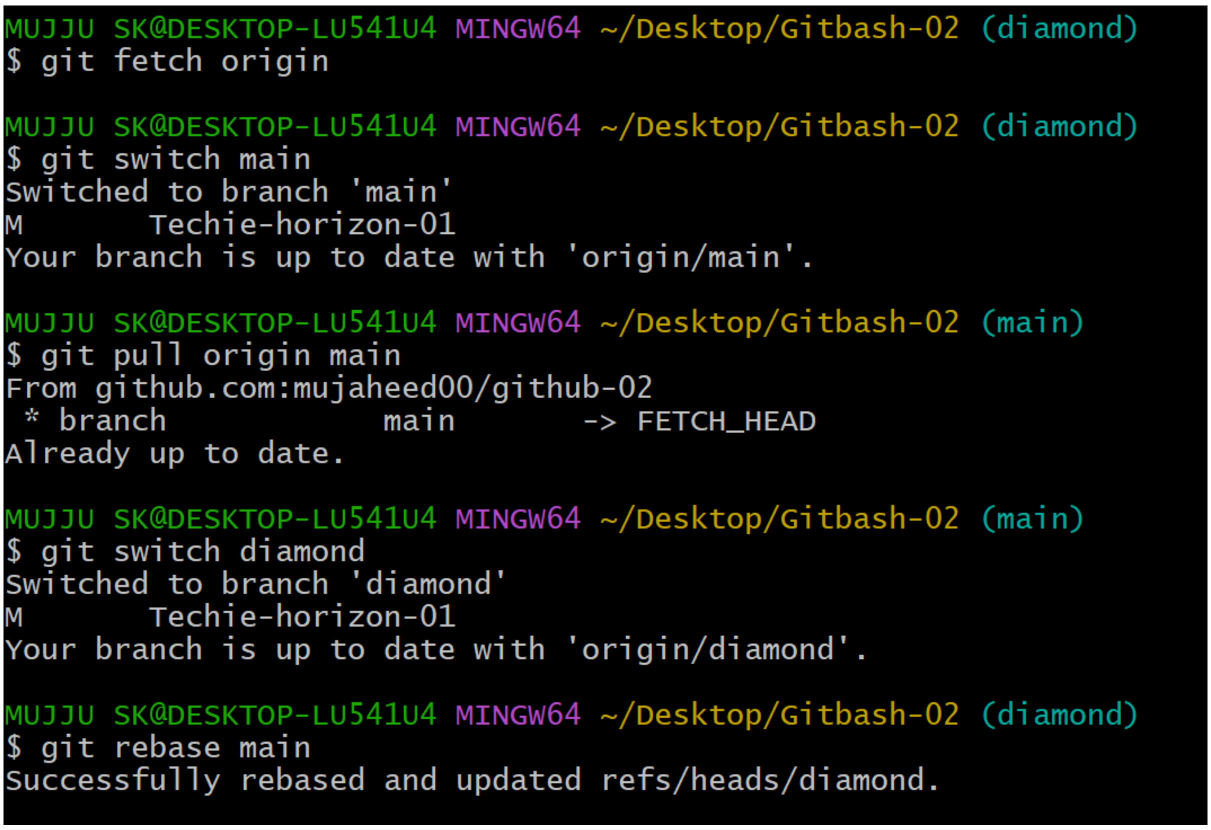


```
MUJJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop/Gitbash-02 (main)
$ git merge --squash diamond
Automatic merge went well; stopped before committing as requested
Squash commit -- not updating HEAD

MUJJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop/Gitbash-02 (main)
$ git commit -m "squashed changes from branch diamond"
On branch main
Your branch is up to date with 'origin/main'.
```

A **squash merge** means you take **all the commits from one branch** and **combine them into a single commit** before merging them into the target branch (like main).

Rebase merge:

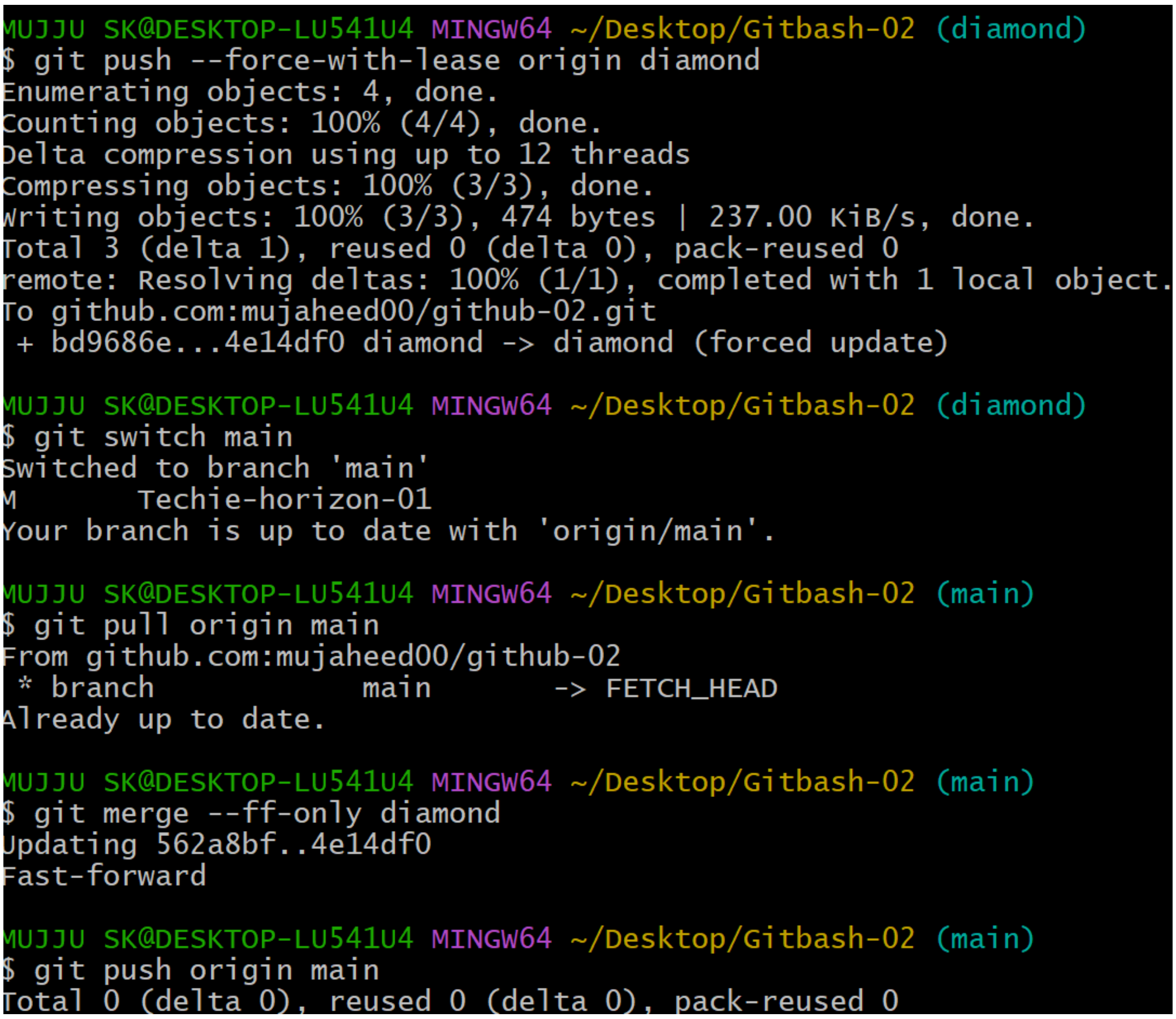
Use the command git fetch origin and switch to main branch and pull main. Use git rebase main to do the merge to main.



```
MUJJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop/Gitbash-02 (diamond)
$ git fetch origin

MUJJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop/Gitbash-02 (diamond)
$ git switch main
Switched to branch 'main'
M       Techie-horizon-01
Your branch is up to date with 'origin/main'.

MUJJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop/Gitbash-02 (main)
$ git pull origin main
From github.com:mujaheed00/github-02
 * branch            main       -> FETCH_HEAD
Already up to date.

MUJJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop/Gitbash-02 (main)
$ git switch diamond
Switched to branch 'diamond'
M       Techie-horizon-01
Your branch is up to date with 'origin/diamond'.

MUJJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop/Gitbash-02 (diamond)
$ git rebase main
Successfully rebased and updated refs/heads/diamond.
```

Use git push --force-with-lease origin diamond command to --force-with-lease is safer than --force because it refuses to overwrite remote changes you don’t have locally.

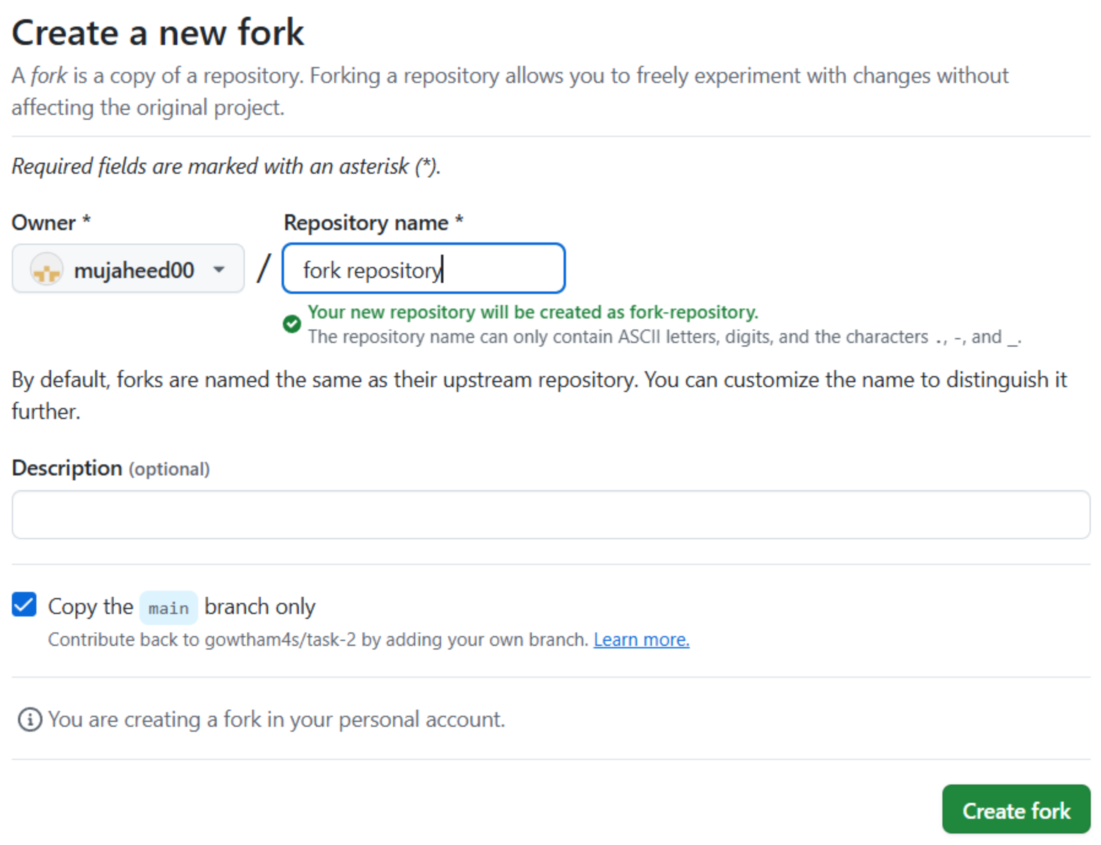
Switch to main and pull the changes. execute the command git merge --ff-only diamond for fast forward after that push to main.



```
MUJJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop/Gitbash-02 (diamond)
$ git push --force-with-lease origin diamond
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 12 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 474 bytes | 237.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To github.com:mujaheed00/github-02.git
 + bd9686e...4e14df0 diamond -> diamond (forced update)

MUJJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop/Gitbash-02 (diamond)
$ git switch main
Switched to branch 'main'
M       Techie-horizon-01
Your branch is up to date with 'origin/main'.

MUJJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop/Gitbash-02 (main)
$ git pull origin main
From github.com:mujaheed00/github-02
 * branch            main       -> FETCH_HEAD
Already up to date.

MUJJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop/Gitbash-02 (main)
$ git merge --ff-only diamond
Updating 562a8bf..4e14df0
Fast-forward

MUJJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop/Gitbash-02 (main)
$ git push origin main
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
```

13. Fork & Pull Request Workflow

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



Create a new fork

A *fork* is a copy of a repository. Forking a repository allows you to freely experiment with changes without affecting the original project.

Required fields are marked with an asterisk (*).

Owner * mujaheed00 / Repository name * fork repository

Your new repository will be created as fork-repository.
The repository name can only contain ASCII letters, digits, and the characters ., -, and _.

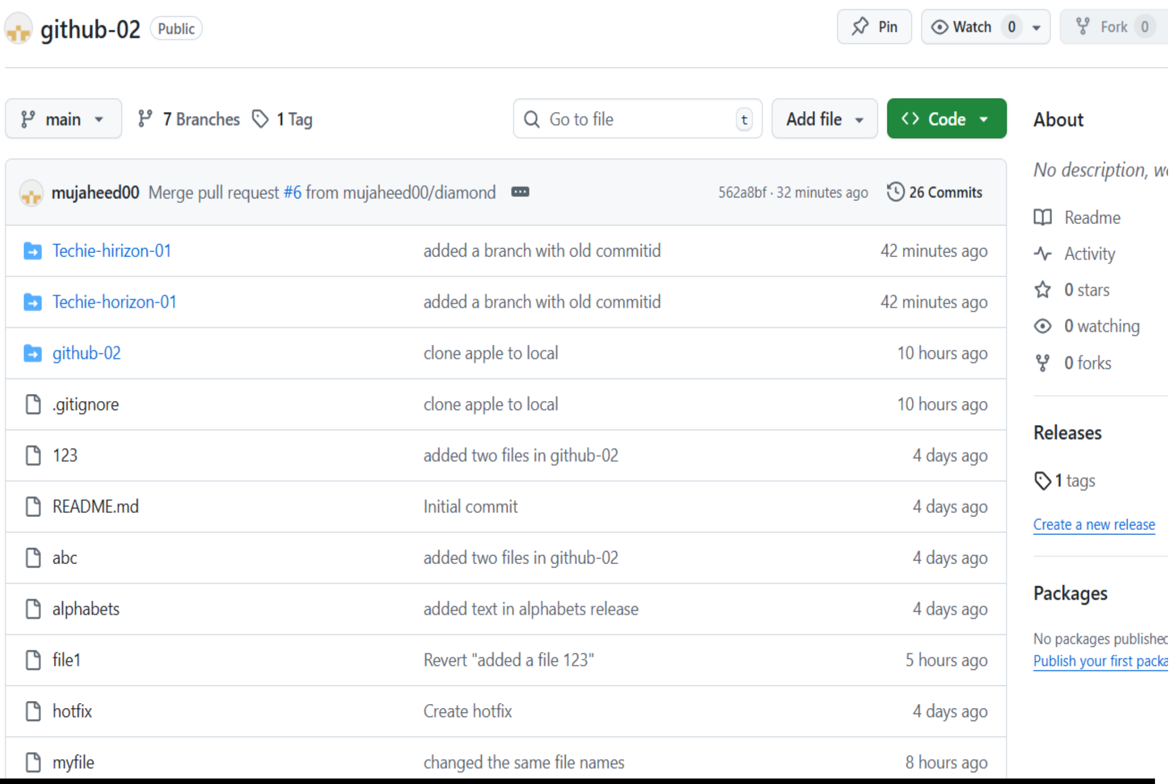
By default, forks are named the same as their upstream repository. You can customize the name to distinguish it further.

Description (optional)

[✓] Copy the `main` branch only
Contribute back to gowtham4s/task-2 by adding your own branch. Learn more.

You are creating a fork in your personal account.

By taking another person's repo URL you need to fork to your account


github-02 Public

Pin | Watch 0 | Fork 0

main | 7 Branches | 1 Tag | Go to file | Add file | Code

mujaheed00 Merge pull request #6 from mujaheed00/diamond · 562a8bf · 32 minutes ago · 26 Commits

Techie-hirizon-01	added a branch with old commitid	42 minutes ago
Techie-horizon-01	added a branch with old commitid	42 minutes ago
github-02	clone apple to local	10 hours ago
.gitignore	clone apple to local	10 hours ago
123	added two files in github-02	4 days ago
README.md	Initial commit	4 days ago
abc	added two files in github-02	4 days ago
alphabets	added text in alphabets release	4 days ago
file1	Revert "added a file 123"	5 hours ago
hotfix	Create hotfix	4 days ago
myfile	changed the same file names	8 hours ago

About

No description, w

Readme
Activity
0 stars
0 watching
0 forks

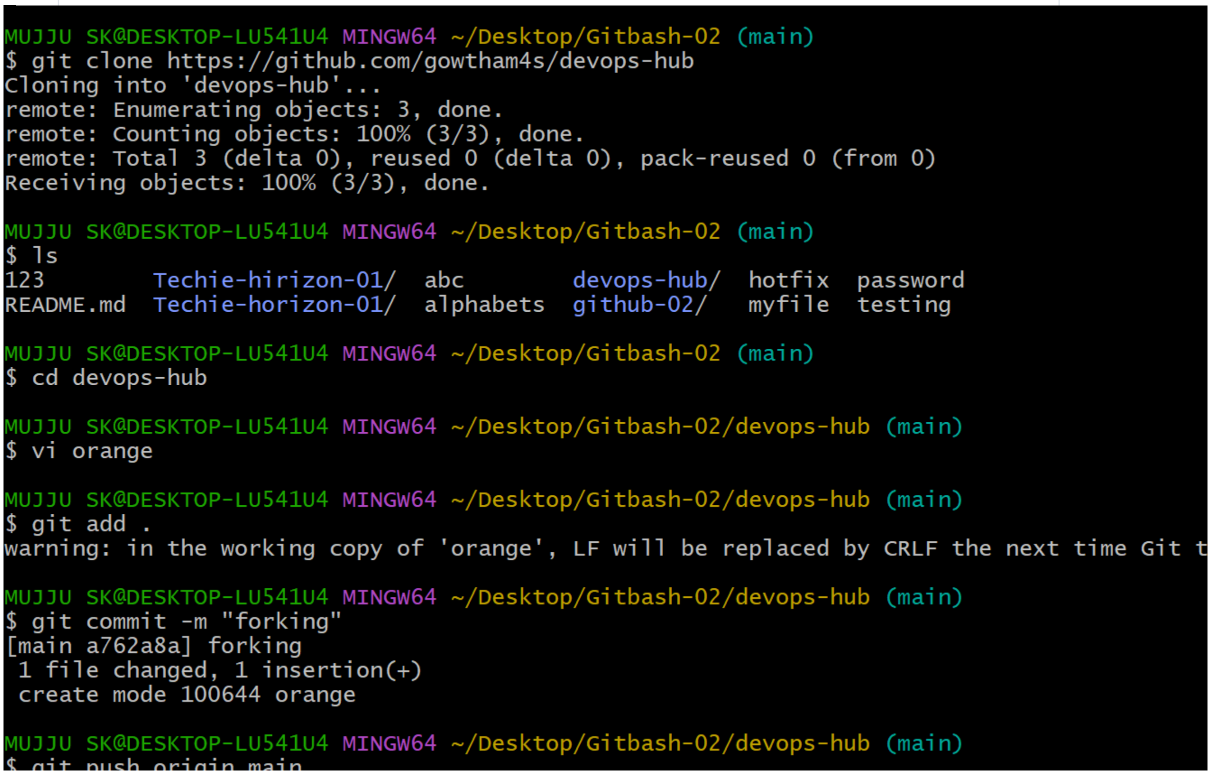
Releases

1 tags

Create a new release

Packages

No packages publishec
Publish your first packa

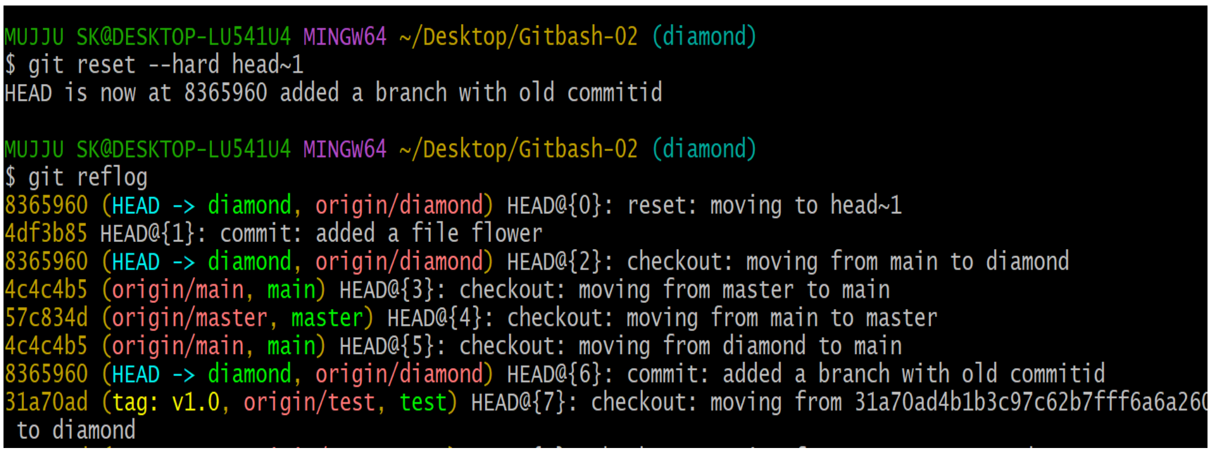

```
MUJJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop/Gitbash-02 (main)
$ git clone https://github.com/gowtham4s/devops-hub
Cloning into 'devops-hub'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (3/3), done.

MUJJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop/Gitbash-02 (main)
$ ls
123        Techie-hirizon-01/  abc        devops-hub/  hotfix  password
README.md  Techie-horizon-01/  alphabets  github-02/   myfile  testing

MUJJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop/Gitbash-02 (main)
$ cd devops-hub

MUJJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop/Gitbash-02/devops-hub (main)
$ vi orange

MUJJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop/Gitbash-02/devops-hub (main)
$ git add .
warning: in the working copy of 'orange', LF will be replaced by CRLF the next time Git t

MUJJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop/Gitbash-02/devops-hub (main)
$ git commit -m "forking"
[main a762a8a] forking
 1 file changed, 1 insertion(+)
 create mode 100644 orange

MUJJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop/Gitbash-02/devops-hub (main)
$ git push origin main
```

14. Recover Lost Commit

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

Create a file with the name flower and commit it

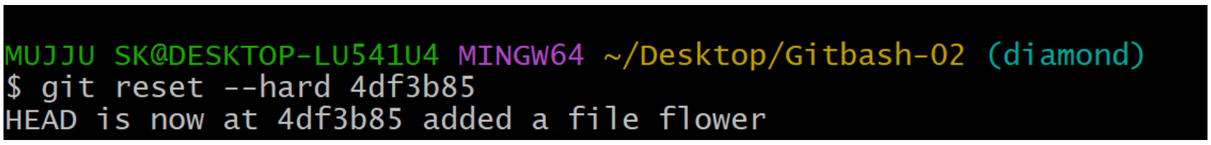
Head is at old commit id .I need to change by using git reflog then it will show all the commits with commit id’s.



```
MUJJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop/Gitbash-02 (diamond)
$ git reset --hard head~1
HEAD is now at 8365960 added a branch with old commitid

MUJJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop/Gitbash-02 (diamond)
$ git reflog
8365960 (HEAD -> diamond, origin/diamond) HEAD@{0}: reset: moving to head~1
4df3b85 HEAD@{1}: commit: added a file flower
8365960 (HEAD -> diamond, origin/diamond) HEAD@{2}: checkout: moving from main to diamond
4c4c4b5 (origin/main, main) HEAD@{3}: checkout: moving from master to main
57c834d (origin/master, master) HEAD@{4}: checkout: moving from main to master
4c4c4b5 (origin/main, main) HEAD@{5}: checkout: moving from diamond to main
8365960 (HEAD -> diamond, origin/diamond) HEAD@{6}: commit: added a branch with old commitid
31a70ad (tag: v1.0, origin/test, test) HEAD@{7}: checkout: moving from 31a70ad4b1b3c97c62b7fff6a6a26
 to diamond
```

Then use git reset command to do the head to our commit

Use command git reset –hard commitid.



```
MUJJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop/Gitbash-02 (diamond)
$ git reset --hard 4df3b85
HEAD is now at 4df3b85 added a file flower
```