

ASSIGNMENT 1 DevOps

IT_BE_41_Himanshu Shukla

Perform the following operations on Git and GitHub.

- Create a Repository
- Perform Add, Commit, Push, Pull
- Create Branches
- Fork a project, Open and merge Pull request
- Merge and Rebase
- Squashing Commits
- Delete branches
- Undo commits

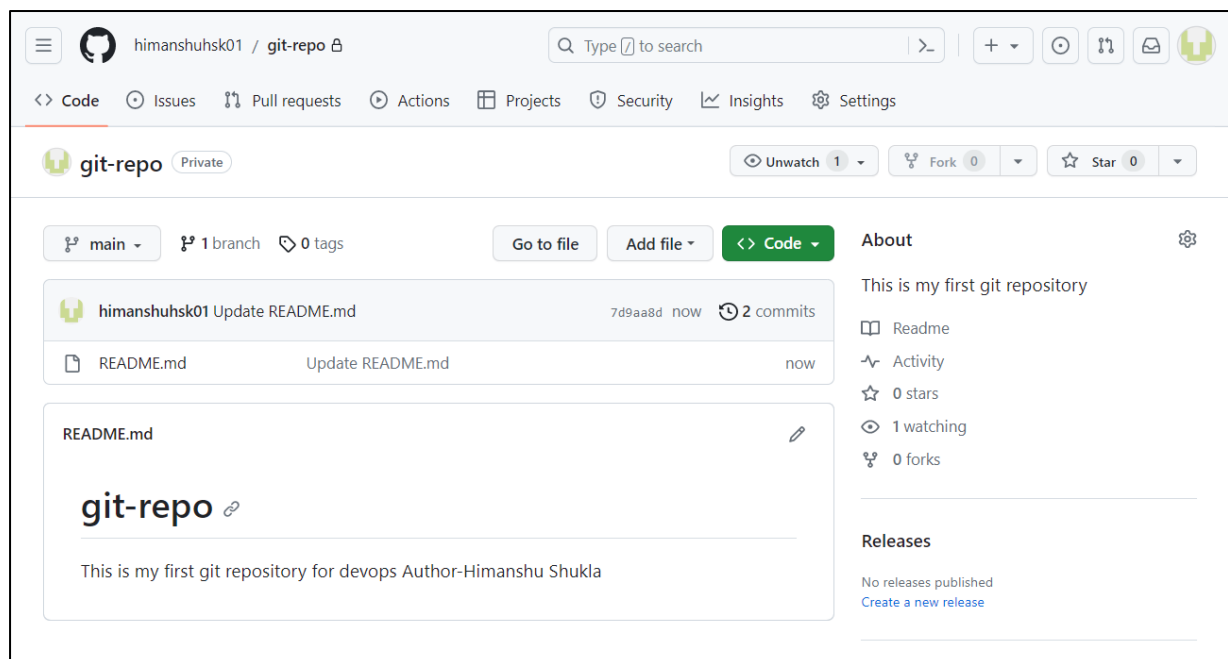
Prerequisite: git-scm, github account, VS code

For performing git operation we can run the git commands either on git bash or VS code terminal

1)Creating a repository:

creating a git repository on **github** and accessing it from remote to local machine

a)create a git repository on github



b) accessing it to local machine

create a new folder, open with VS code then open vs code terminal and write the git commands below

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

Copyright (C) 2014 Microsoft Corporation. All rights reserved.

PS E:\project\git repo> git --version
git version 2.39.0.windows.2
PS E:\project\git repo> git clone https://github.com/himanshuhs01/git-repo.git
Cloning into 'git-repo'...
info: please complete authentication in your browser
...
remote: Enumerating objects: 6, done.
remote: Counting objects: 100% (6/6), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 6 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (6/6), done.
PS E:\project\git repo> 
```

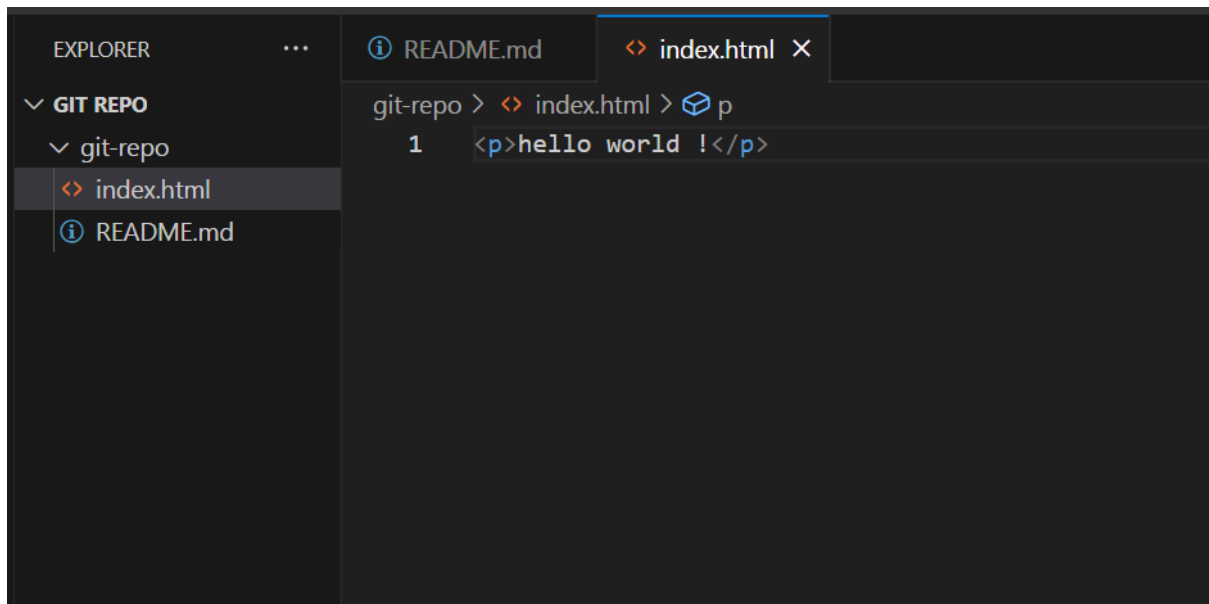
Here the same repo of github has been cloned on local machine

```
EXPLORER  ...  README.md x
v GIT_REPO [+] [?] [?] [?]
v git-repo
  README.md

git-repo > README.md > # git-repo
1 # git-repo
2 This is my first git repository for devops
3 Author-Himanshu Shukla
4 
```

2)Performing ADD COMMIT PUSH & PULL

Before starting with operation just change the directory to git-repo using `cd git-repo`
create a new file named index.html in same folder of git –repo



a)add, commit: This operation will help in saving the modified file on local machine

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  powershell +
PS E:\project\git repo\git-repo> git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   index.html

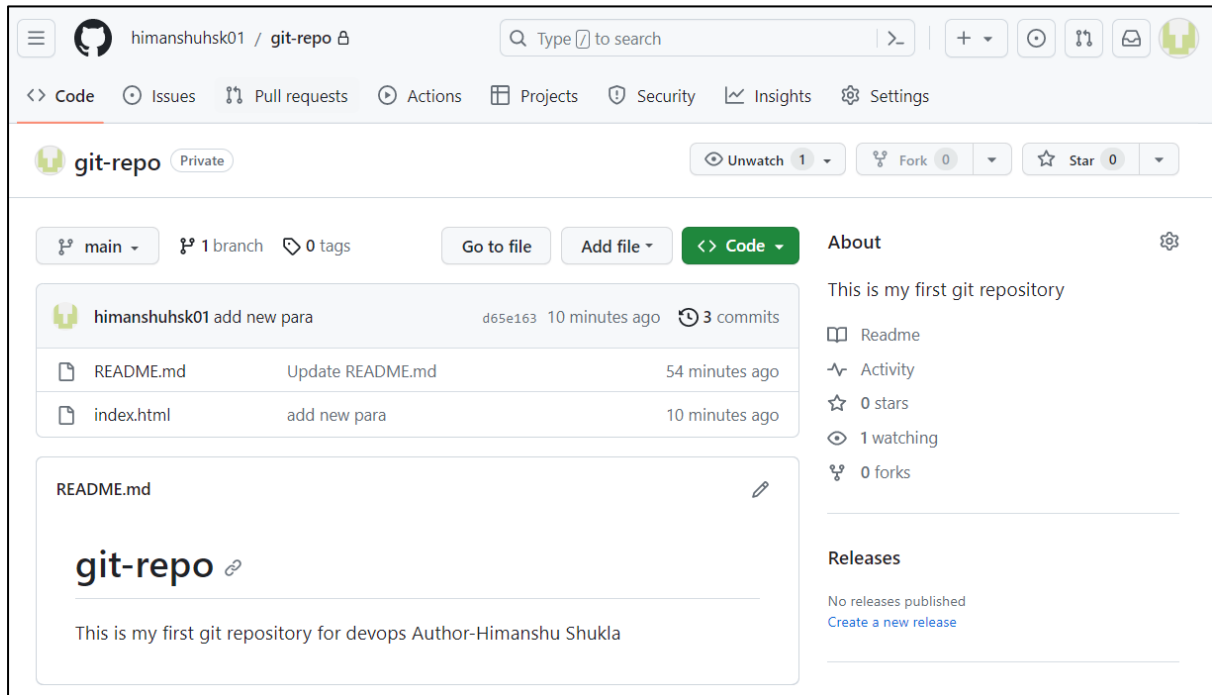
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   index.html

PS E:\project\git repo\git-repo> git add .
PS E:\project\git repo\git-repo> git commit -m "add new para"
[main d65e163] add new para
1 file changed, 1 insertion(+)
create mode 100644 index.html
```

b)push: This operation will reflect the changes on github repo(remote) since uptill commit it was saved only on local machine

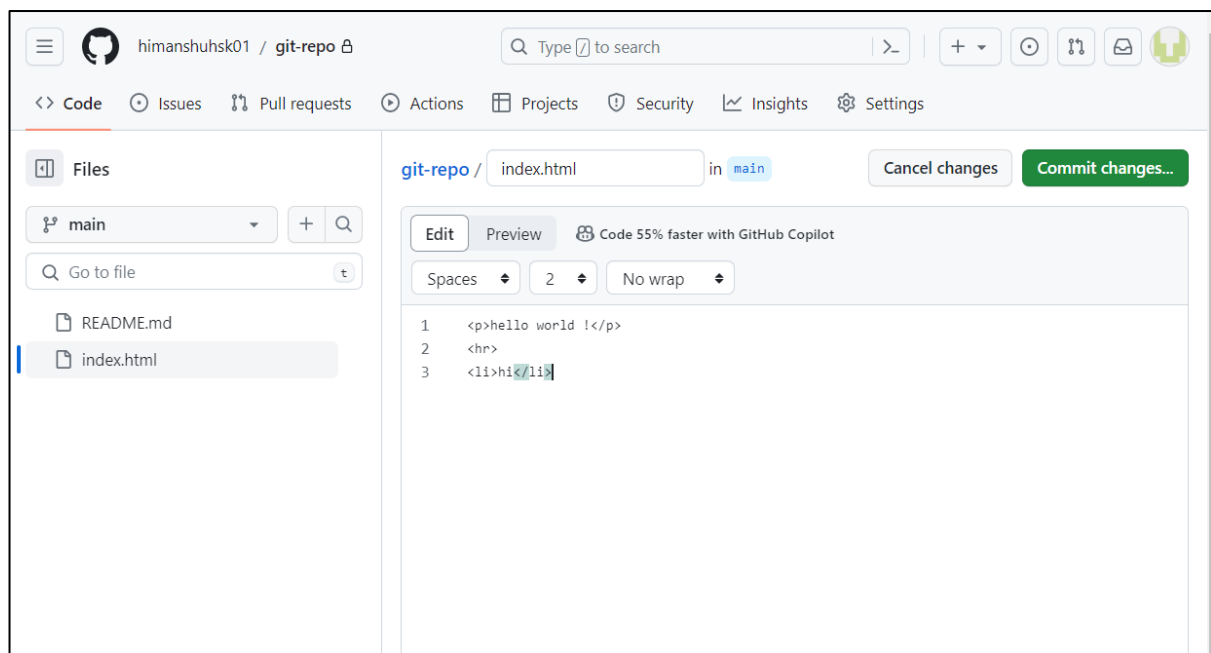
```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  powershell
PS E:\project\git repo\git-repo> git push origin main
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 307 bytes | 307.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/himanshuhsk01/git-repo.git
7d9aa8d..d65e163  main -> main
```

After this we can see that our newly created file index.html has been pushed on github

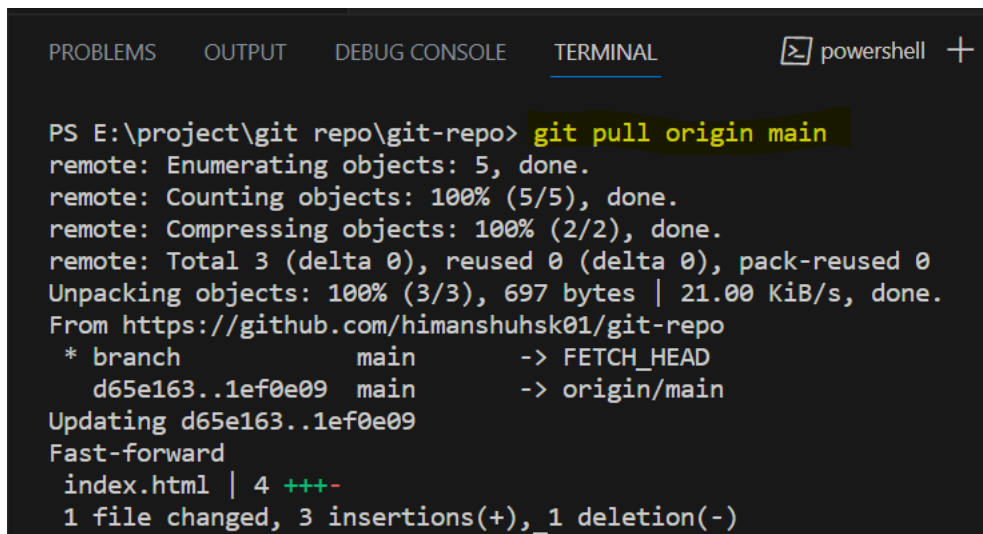


c)PULL: It is used to immediately update the content on local machine repo from remote github repo

now suppose I have modified the index file by adding few more lines so let see it ,

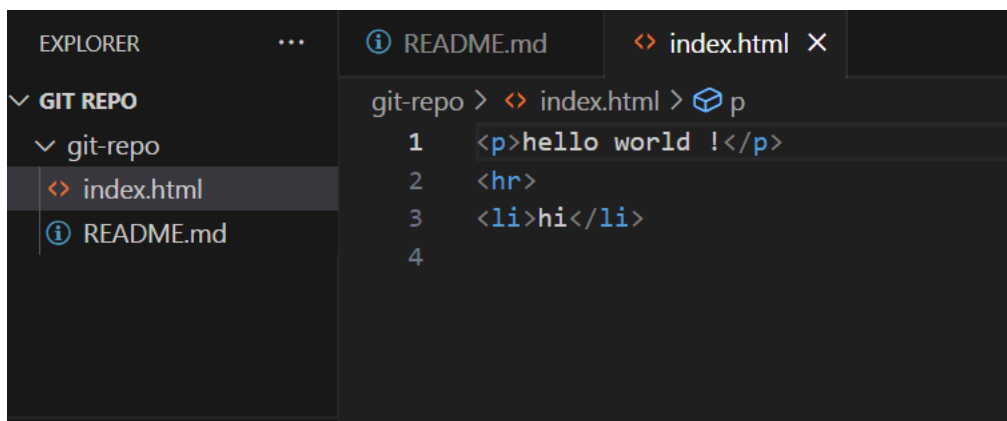


Now again open terminal and write the command



```
PS E:\project\git repo\git-repo> git pull origin main
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 697 bytes | 21.00 KiB/s, done.
From https://github.com/himanshuhs01/git-repo
* branch      main      -> FETCH_HEAD
d65e163..1ef0e09 main    -> origin/main
Updating d65e163..1ef0e09
Fast-forward
 index.html | 4 +++-
 1 file changed, 3 insertions(+), 1 deletion(-)
```

Now the update can be clearly visible in local machine(your desktop) repo

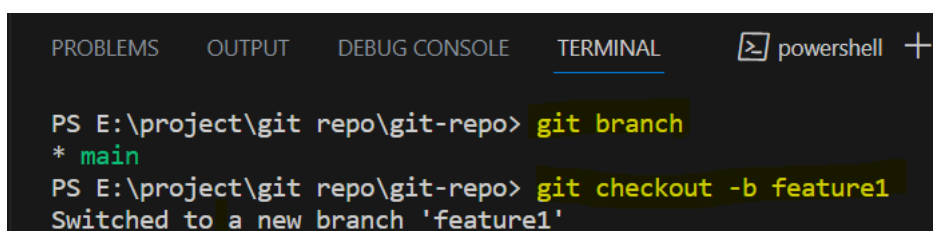


```
git-repo > < index.html > p
1  <p>hello world !</p>
2  <hr>
3  <li>hi</li>
4
```

3) Create Branches

Before creating branch check that the branch name is “main” if not change it to “main” using command `git branch -M main`

a) create a new branch named feature1



```
PS E:\project\git repo\git-repo> git branch
* main
PS E:\project\git repo\git-repo> git checkout -b feature1
Switched to a new branch 'feature1'
```

b) Modify branch(feature1) of main

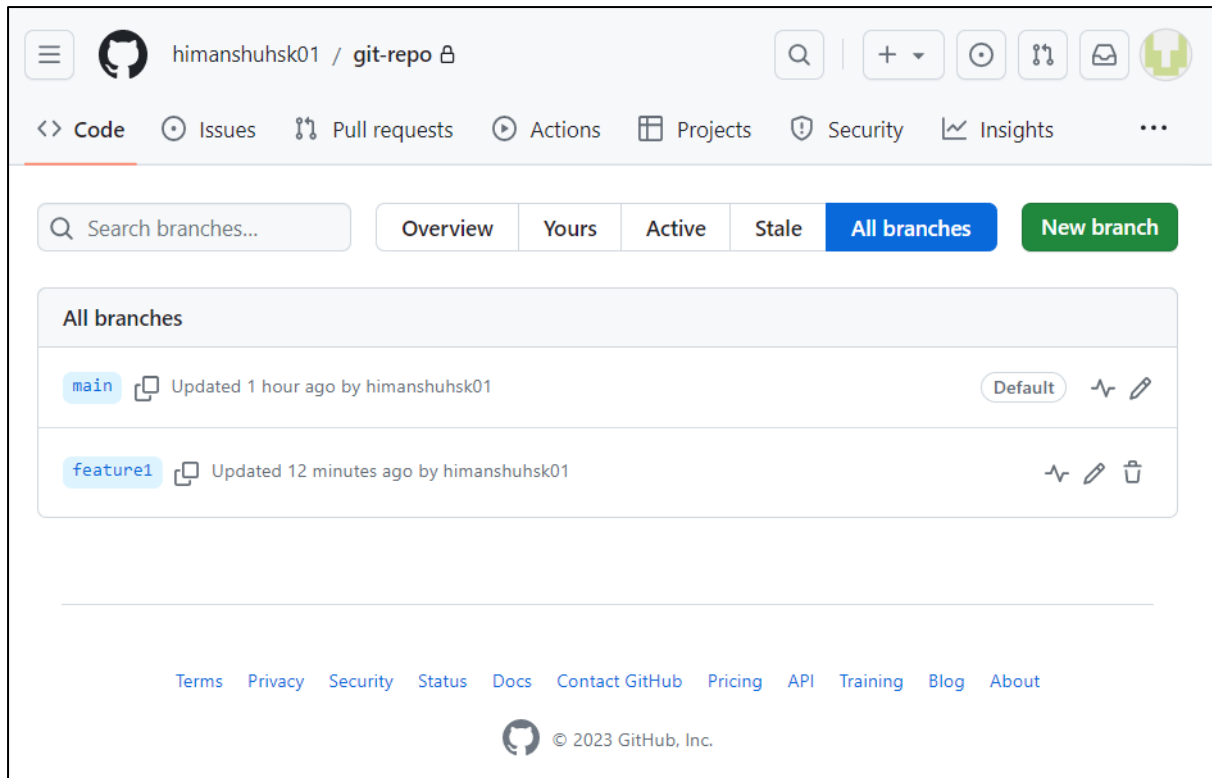


After this, push it to github repo using git commands where you can see the two branches now

```
PS E:\project\git repo\git-repo> git status
On branch feature1
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:   index.html

no changes added to commit (use "git add" and/or "git commit -a")
PS E:\project\git repo\git-repo> git add .
PS E:\project\git repo\git-repo> git commit -m "add new feature1"
[feature1 6da36d2] add new feature1
 1 file changed, 1 insertion(+)
PS E:\project\git repo\git-repo> git push origin feature1
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 353 bytes | 353.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'feature1' on GitHub by visiting:
remote:   https://github.com/himanshuhs01/git-repo/pull/new/feature1
remote:
To https://github.com/himanshuhs01/git-repo.git
 * [new branch]      feature1 -> feature1
```

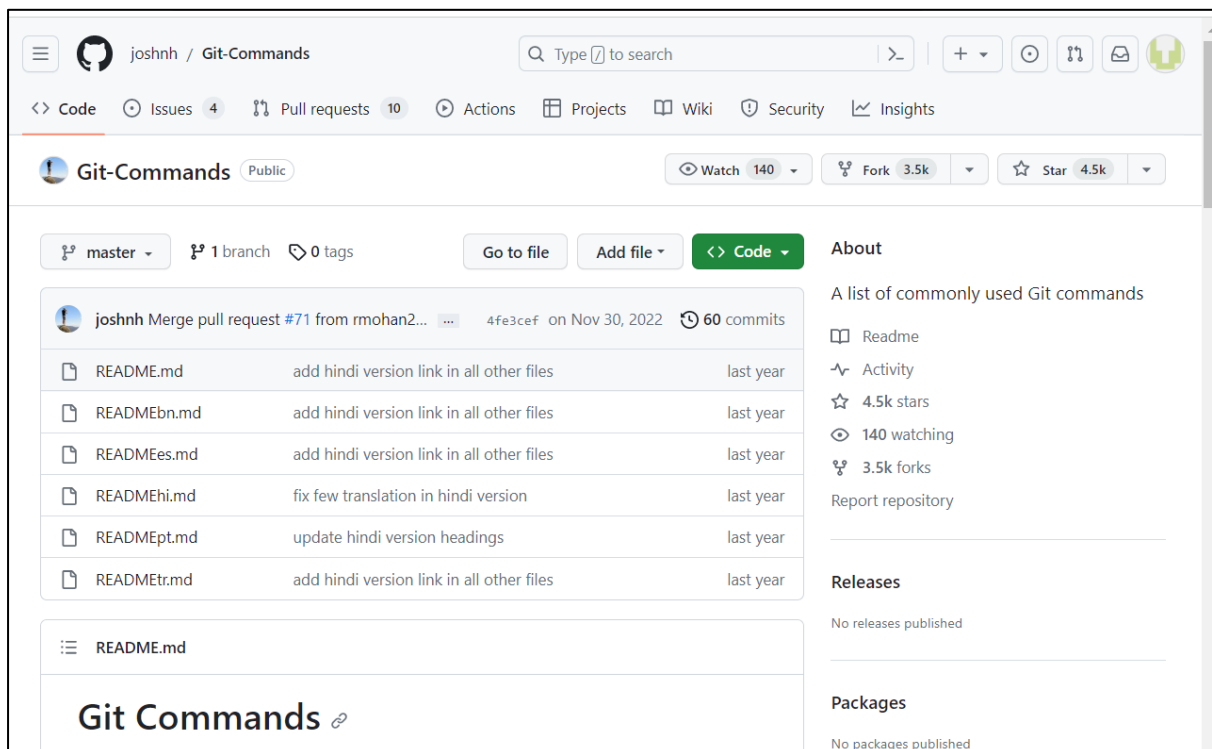
Now we can see that the two branches has been formed



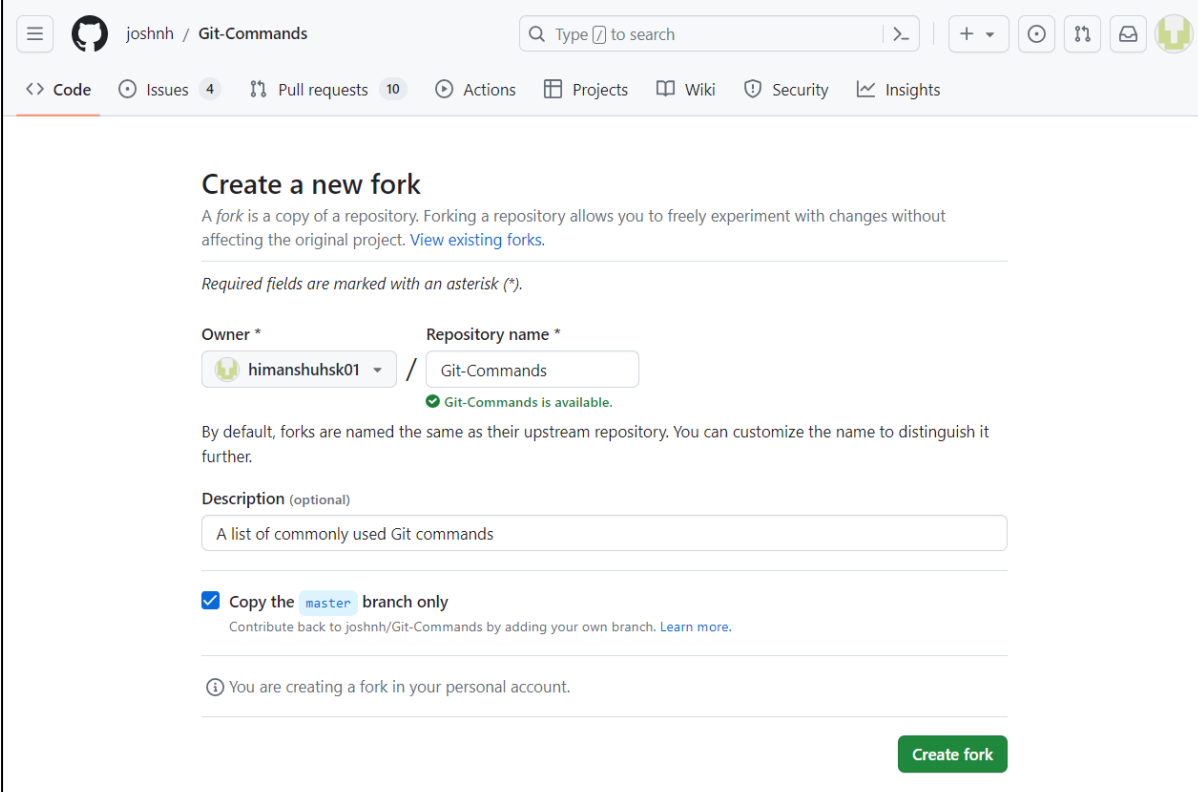
4) Fork a project, Open and merge Pull request

a) forking a random project of joshnh “Git-Commands”

This is Joshnh account, now we will fork it to our account



Now after clicking on create fork this whole branch will be copied to our account



The screenshot shows the GitHub interface for creating a new fork of the repository 'Git-Commands' by user 'joshnh'. The page title is 'Create a new fork'. Below the title, there is a brief explanation of forking: 'A fork is a copy of a repository. Forking a repository allows you to freely experiment with changes without affecting the original project. [View existing forks.](#)'. A note states 'Required fields are marked with an asterisk (*)'. The form has two main sections: 'Owner *' and 'Repository name *'. The 'Owner *' section has a dropdown menu showing 'himanshuhs01'. The 'Repository name *' section has a text input field containing 'Git-Commands'. Below these fields, a green checkmark indicates 'Git-Commands is available.'. A note explains that forks are named the same as their upstream repository by default. The 'Description (optional)' section has a text input field containing 'A list of commonly used Git commands'. Below the description, there is a checkbox labeled 'Copy the master branch only' which is checked. A note below the checkbox says 'Contribute back to joshnh/Git-Commands by adding your own branch. [Learn more.](#)'. At the bottom, there is an information icon and a note: 'You are creating a fork in your personal account.'. A green 'Create fork' button is located at the bottom right of the form.

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. [View existing forks.](#)

Required fields are marked with an asterisk (*).

Owner * Repository name *

himanshuhs01 / Git-Commands

Git-Commands is available.

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

Description (optional)

A list of commonly used Git commands

☒ Copy the master branch only

Contribute back to joshnh/Git-Commands by adding your own branch. [Learn more.](#)

You are creating a fork in your personal account.

Create fork

b) Open and merge pull request

now will merge our modified feature1 branch with main branch using pull request and then will pull it to our local machine

Now go to github account click on [compare and pull request](#) on top

add new feature1 #1



himanshuhs01 wants to merge 1 commit into `main` from `feature1`



Conversation 0



Commits 1



Checks 0



Files changed 1



himanshuhs01 commented now

...

No description provided.



add new feature1

6da36d2

Add more commits by pushing to the `feature1` branch on `himanshuhs01/git-repo`.



Require approval from specific reviewers before merging

Branch protection rules ensure specific people approve pull requests before they're merged.

Add rule

×



Continuous integration has not been set up

GitHub Actions and several other apps can be used to automatically catch bugs and enforce style.



This branch has no conflicts with the base branch

Merging can be performed automatically.

Merge pull request



You can also [open this in GitHub Desktop](#) or view [command line instructions](#).

Now the branch has been successfully merged

add new feature1 #1



himanshuhs01 merged 1 commit into `main` from `feature1` now



Conversation 0



Commits 1



Checks 0



Files changed 1



himanshuhs01 commented 4 minutes ago

...

No description provided.



add new feature1

6da36d2



himanshuhs01 merged commit b3ad57c into `main` now

Revert



Pull request successfully merged and closed

You're all set—the `feature1` branch can be safely deleted.

Delete branch

Pull it to local machine using command `git pull origin main`

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
[>] powershell + ▢ 🗑️ ⋮

PS E:\project\git repo\git-repo> git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
PS E:\project\git repo\git-repo> git pull origin main
remote: Enumerating objects: 1, done.
remote: Counting objects: 100% (1/1), done.
remote: Total 1 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (1/1), 634 bytes | 57.00 KiB/s, done.
From https://github.com/himanshuhs01/git-repo
* branch                main                -> FETCH_HEAD
   1ef0e09..b3ad57c      main                -> origin/main
Updating 1ef0e09..b3ad57c
Fast-forward
 index.html | 1 +
 1 file changed, 1 insertion(+)

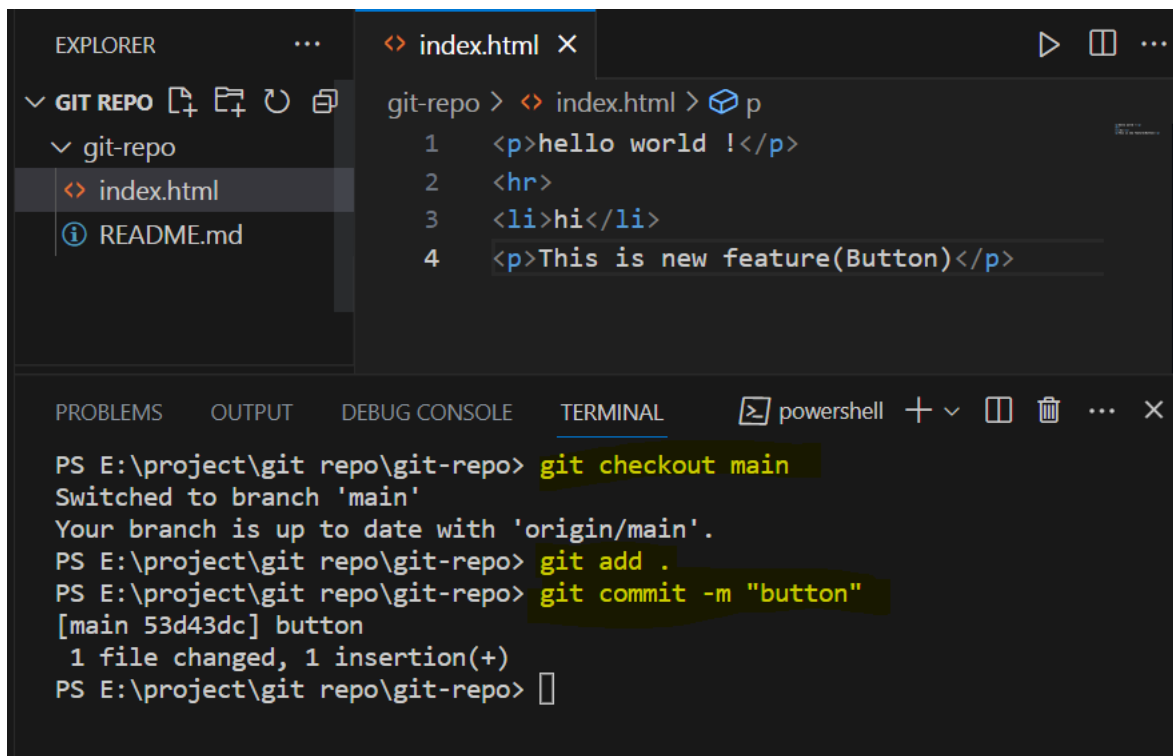
```

Now after this the merging can be seen in local machine repo too.

5) Merge and Rebase

Now some time there will be a conflict merge so let see how to merge the conflict content

Modify the same line on both branches main and feature1



EXPLORER

- ✓ GIT REPO
 - git-repo
 - index.html
 - README.md

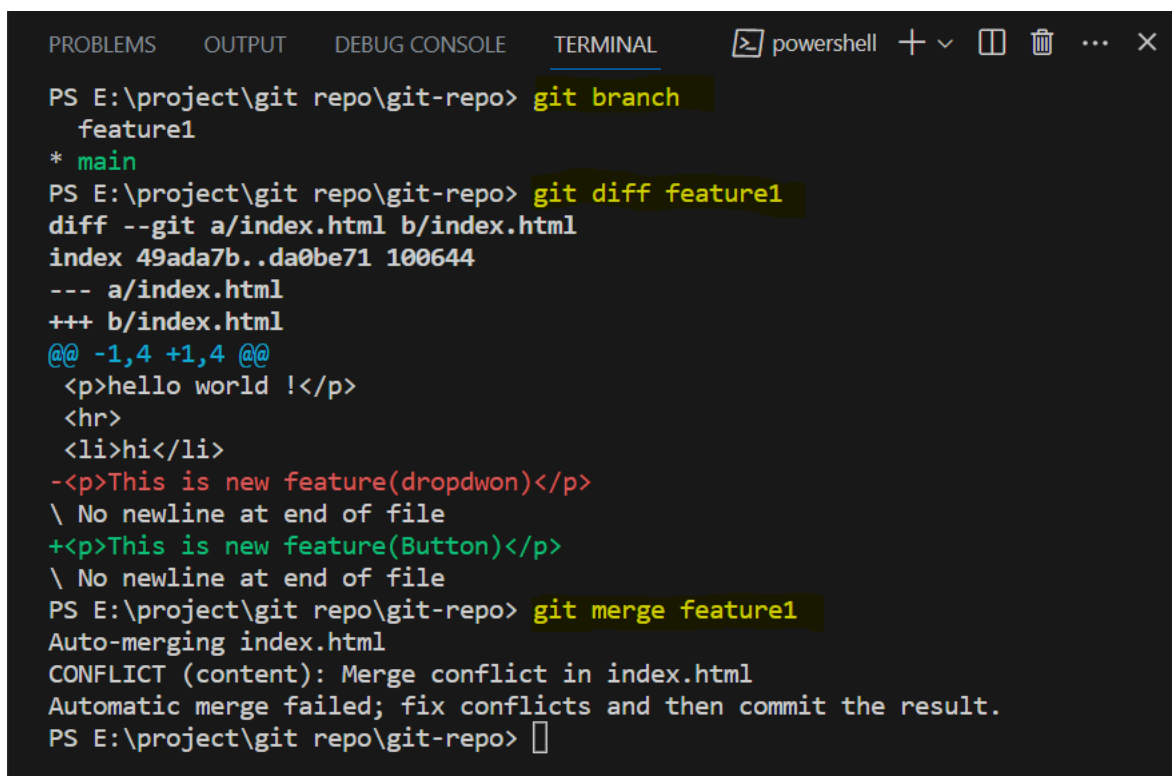
index.html

```
git-repo > <> index.html > p
1 <p>hello world !</p>
2 <hr>
3 <li>hi</li>
4 <p>This is new feature(Button)</p>
```

TERMINAL

```
PS E:\project\git repo\git-repo> git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
PS E:\project\git repo\git-repo> git add .
PS E:\project\git repo\git-repo> git commit -m "button"
[main 53d43dc] button
1 file changed, 1 insertion(+)
PS E:\project\git repo\git-repo>
```

Now will merge this conflict and will vanish one of line of branches



TERMINAL

```
PS E:\project\git repo\git-repo> git branch
feature1
* main
PS E:\project\git repo\git-repo> git diff feature1
diff --git a/index.html b/index.html
index 49ada7b..da0be71 100644
--- a/index.html
+++ b/index.html
@@ -1,4 +1,4 @@
<p>hello world !</p>
<hr>
<li>hi</li>
- <p>This is new feature(dropdwon)</p>
\ No newline at end of file
+ <p>This is new feature(Button)</p>
\ No newline at end of file
PS E:\project\git repo\git-repo> git merge feature1
Auto-merging index.html
CONFLICT (content): Merge conflict in index.html
Automatic merge failed; fix conflicts and then commit the result.
PS E:\project\git repo\git-repo>
```

```
git-repo > <> index.html > ?
1 <p>hello world !</p>
2 <hr>
3 <li>hi</li>
4 <<<<<< HEAD (Current Change)
5 <p>This is new feature(Button)</p>
6 =====
7 <p>This is new feature(dropdwon)</p>
8 >>>>>> feature1 (Incoming Change)
9
```

Now just delete the unwanted thing and merge it after this push to main github page

```
git-repo > <> index.html > ...
1 <p>hello world !</p>
2 <hr>
3 <li>hi</li>
4 <p>This is new feature(Button)</p>
5 <p>This is new feature(dropdwon)</p>
6
7
```

now write the commad

```
PS E:\project\git repo\git-repo> git add .
PS E:\project\git repo\git-repo> git commit -m "Add both features"
[main 74ea3df] Add both features
PS E:\project\git repo\git-repo> git status
On branch main
Your branch is ahead of 'origin/main' by 4 commits.
(use "git push" to publish your local commits)

nothing to commit, working tree clean
PS E:\project\git repo\git-repo> git checkout feature1
Switched to branch 'feature1'
PS E:\project\git repo\git-repo> git merge main
Updating 509a8c7..74ea3df
Fast-forward
 index.html | 4 +++-
 1 file changed, 3 insertions(+), 1 deletion(-)
```

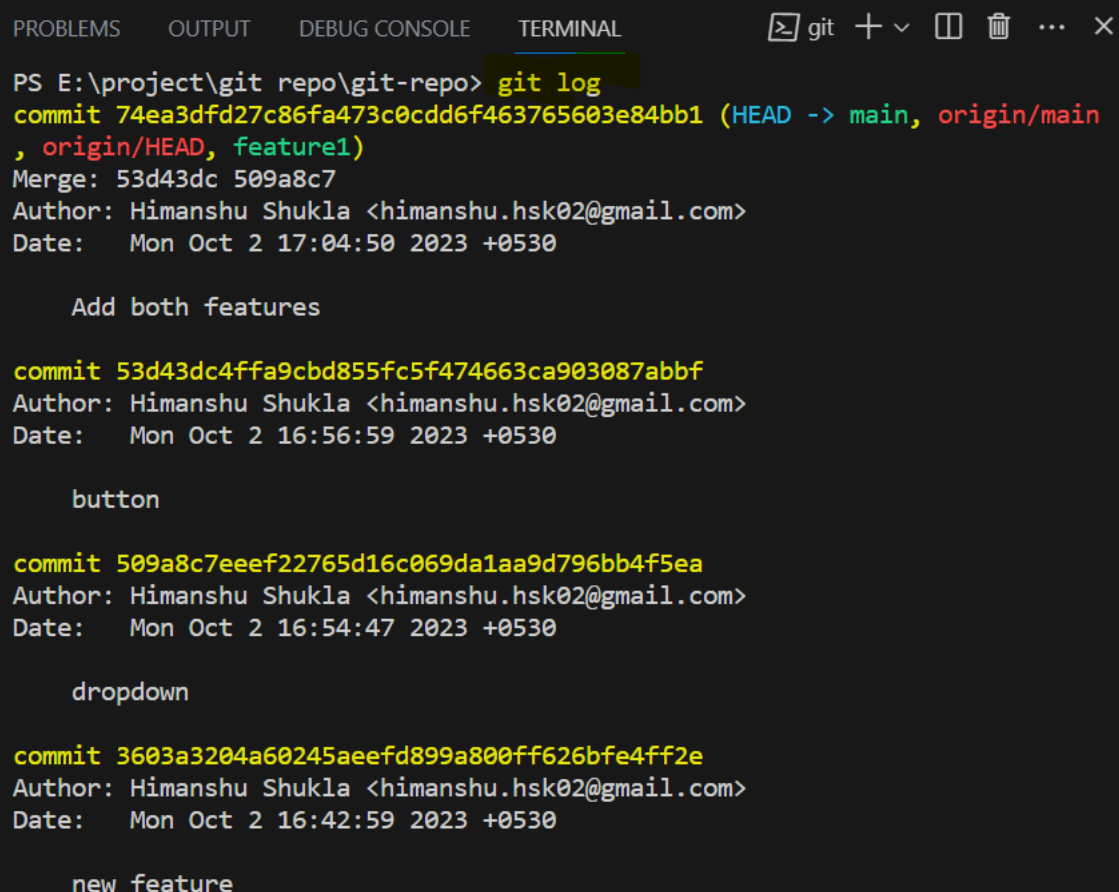
```
PS E:\project\git repo\git-repo> git push --force
Enumerating objects: 29, done.
Counting objects: 100% (29/29), done.
Delta compression using up to 4 threads
Compressing objects: 100% (24/24), done.
Writing objects: 100% (29/29), 4.37 KiB | 2.19 MiB/s, done.
Total 29 (delta 4), reused 16 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (4/4), done.
To https://github.com/himanshuhs01/git-repo.git
+ 56c1783...74ea3df main -> main (forced update)
```

Now after refreshing it will be reflected on main github repo

6) Squashing commits

Squash will retain the commits

Check the commits using `git log`



```
PS E:\project\git repo\git-repo> git log
commit 74ea3dfd27c86fa473c0cdd6f463765603e84bb1 (HEAD -> main, origin/main
, origin/HEAD, feature1)
Merge: 53d43dc 509a8c7
Author: Himanshu Shukla <himanshu.hsk02@gmail.com>
Date: Mon Oct 2 17:04:50 2023 +0530

    Add both features

commit 53d43dc4ffa9cbd855fc5f474663ca903087abbbf
Author: Himanshu Shukla <himanshu.hsk02@gmail.com>
Date: Mon Oct 2 16:56:59 2023 +0530

    button

commit 509a8c7eeef22765d16c069da1aa9d796bb4f5ea
Author: Himanshu Shukla <himanshu.hsk02@gmail.com>
Date: Mon Oct 2 16:54:47 2023 +0530

    dropdown

commit 3603a3204a60245aeefd899a800ff626bfe4ff2e
Author: Himanshu Shukla <himanshu.hsk02@gmail.com>
Date: Mon Oct 2 16:42:59 2023 +0530

    new feature
```

Enter q to quite the log

After write the command “git rebase -i HEAD~3” and hit enter, change the pick to s

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
git + - - - x

pick 3603a32 new feature
s 53d43dc button
s 30249b4 add new feature
s 509a8c7 dropdown

# Rebase b3ad57c..74ea3df onto b3ad57c (4 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.
#
.git/rebase-merge/git-rebase-todo[+] [unix] (18:00 02/10/2023) 8,1 Top
:wq
```

After this you can see that the two commits have been merged

```
commit 661f24151f14d95b8d700651439b1888c53795e7 (HEAD)
Author: Himanshu Shukla <himanshu.hsk02@gmail.com>
Date: Mon Oct 2 16:42:59 2023 +0530

# This is a combination of 2 commits.
# This is the 1st commit message:

new feature

# This is the commit message #2:

button

commit b3ad57c893da2ca1d8e63e77e1bf6f0c71288ebd
Merge: 1ef0e09 6da36d2
Author: Himanshu Shukla <121927424+himanshuhs01@users.noreply.github.com>
Date: Mon Oct 2 15:43:12 2023 +0530

Merge pull request #1 from himanshuhs01/feature1

add new feature1

commit 6da36d21c2840809e353ece8d1e12408d509be6a
Author: Himanshu Shukla <himanshu.hsk02@gmail.com>
Date: Mon Oct 2 14:55:11 2023 +0530

add new feature1

commit 1ef0e094f966ca91479c1fa43c128625c6a6df1a
Author: Himanshu Shukla <121927424+himanshuhs01@users.noreply.github.com>
Date: Mon Oct 2 14:05:33 2023 +0530

Update index.html

:
```

7) Delete the branch

Create a new branch and delete it

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  powershell + - [ ] [ ] ... X

PS E:\project\git repo\git-repo> git checkout -b feature
Switched to a new branch 'feature'
PS E:\project\git repo\git-repo> git branch
* feature
  feature1
  main
PS E:\project\git repo\git-repo> git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
PS E:\project\git repo\git-repo> git branch -d feature
Deleted branch feature (was 74ea3df).
PS E:\project\git repo\git-repo> git branch
feature1
* main
PS E:\project\git repo\git-repo> [ ]
```

8) undo commits

suppose I have deleted the last line and perform the add operation but now I have to undo it . let us see ,

```
EXPLORER  ...  <> index.html X  ⓘ README.md

v GIT REPO
v git-repo
  <> index.html
  ⓘ README.md

git-repo > <> index.html > ...
1  <p>hello world !</p>
2  <hr>
3  <li>hi</li>
4  <p>This is new feature(Button)</p>
5  |
6
7
```

