

List of Programs

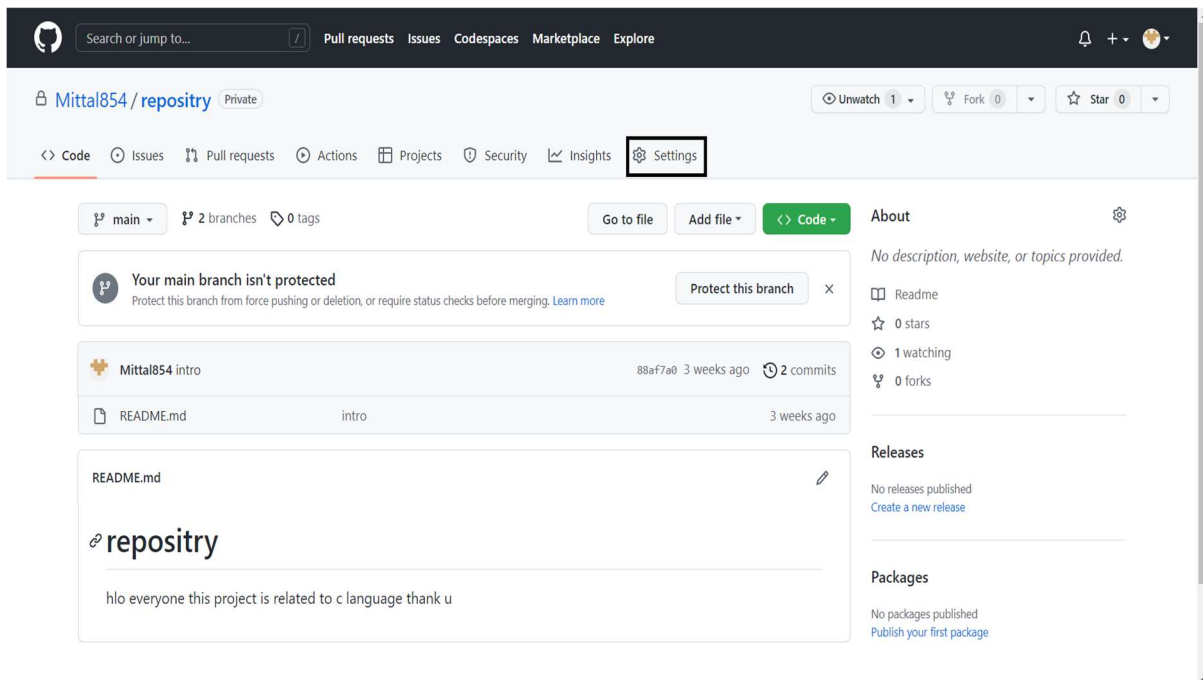
S. No	Program Title	Page No.
1	Setting up of Git Client	3-4
2	Setting up GitHub Account	5-6
3	Generate logs	7
4	Create and visualize branches	8-10
5	Git life cycle description	11-12
6	Add collaborators on GitHub Repo	13-14
7	Fork & commit	15-17
8	Merge and Resolve conflicts created due to own activity and collaborators activity.	18-20
9	Reset and revert	21-24

Experiment 6. Add collaborators to GitHub Repo

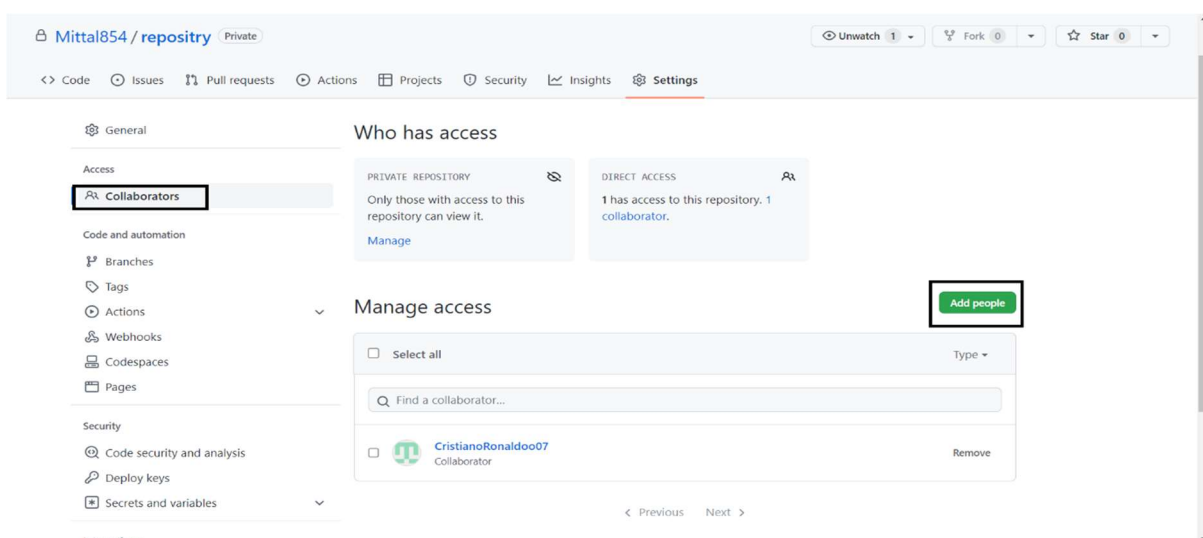
Step 1. Open the GitHub website: <https://github.com/> and log in to your account.

Step 2. Create a new repository.

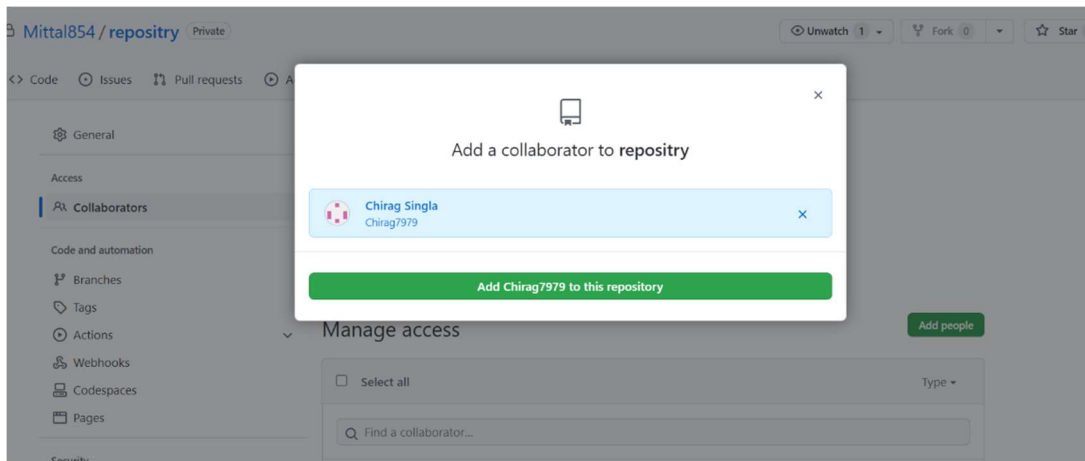
Step 3. After creating a repository, go to settings.



Step 4. Go to collaborators & click on add people.

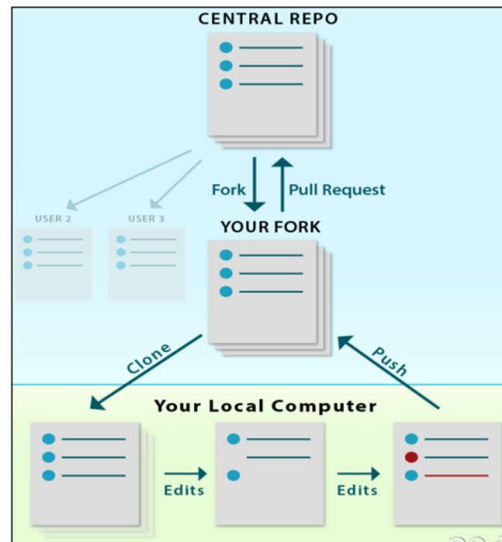


Step 5. Search the collaborators name and add him to the repository.



Experiment 7. Fork & Commit

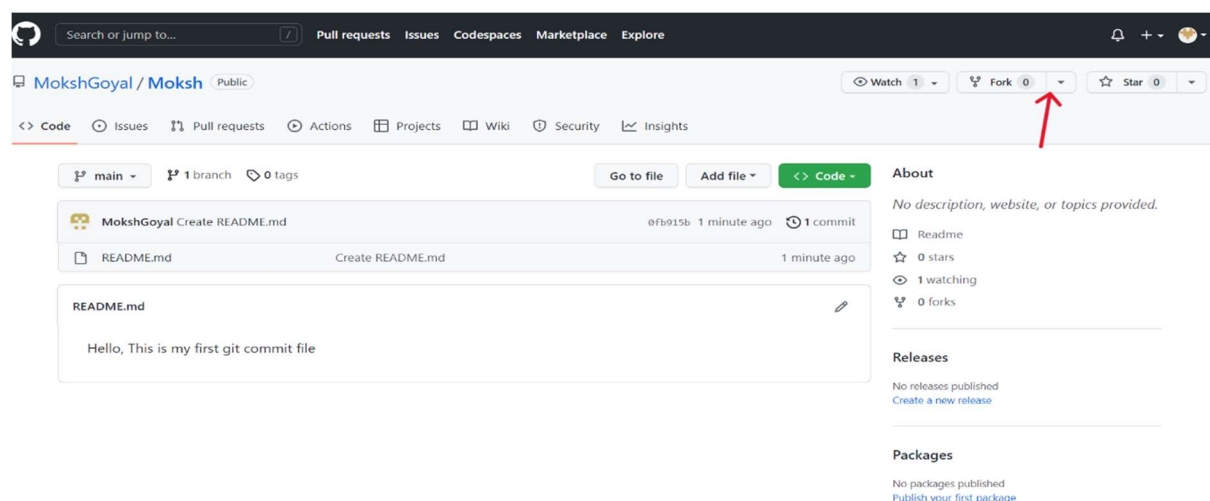
A fork is a copy of a repository that you manage. It allows us to freely experiment with the data. After creating a fork, we can make any desired change like adding collaborators, rename files, generate GitHub pages but all these changes won't be reflected in the original repository.



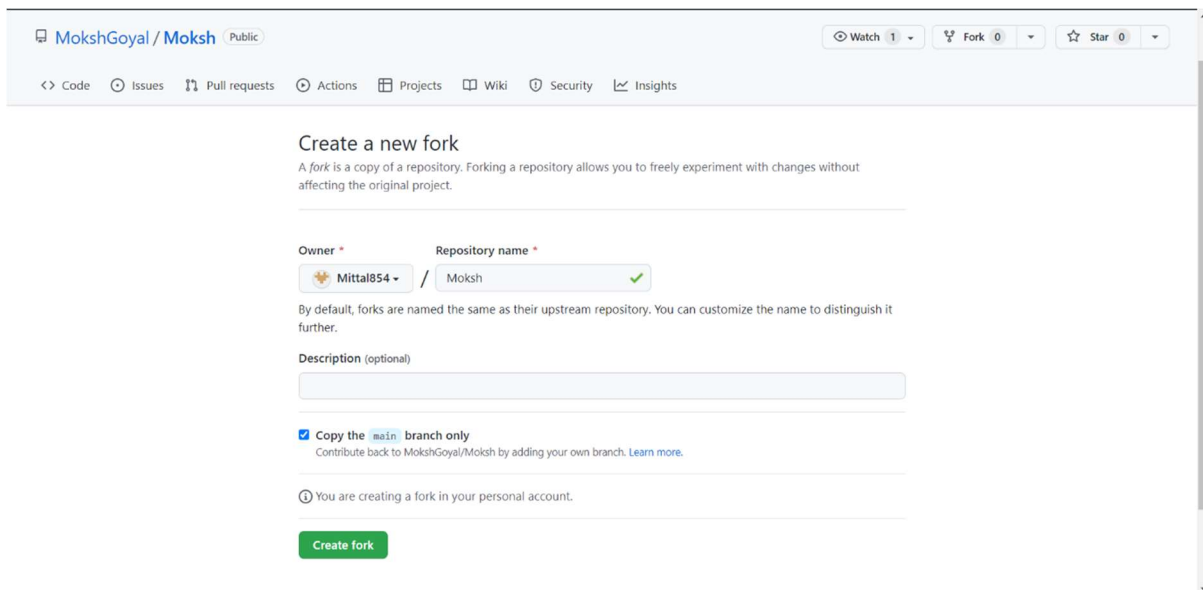
To import the changes into the original repository, the user needs to send a pull request to the maintainer. If the maintainer closes the pull request only then the content can be added to the original repository.

Forking is a better method than directly cloning any repository, as in cloning only the default branch is cloned whereas forking creates a clone of the complete repo and also allows us to push the changes to the main repository by using open and close pull request

Step 1. Open the repository which you want to fork & click on fork option.



Step 2. Click on create fork.



Step 3. A copy of the repo which is forked from other user is formed. We can now make modification without changing main source code.

Step 4. Now use the command **git clone <URL>** to fetch the remote repo or clone the repo.

```

MINGW64:/d/scm-lab2023
bhuve@mittal MINGW64 /d/scm-lab2023 (master)
$ git clone https://github.com/Mittal854/Moksh.git
Cloning into 'Moksh'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (3/3), done.

```

Step 5. Now we can open the file, make changes & commit.

MINGW64:/d/scm-lab2023

```
bhuve@mittal MINGW64 /d/scm-lab2023 (master)
$ git branch fork

bhuve@mittal MINGW64 /d/scm-lab2023 (master)
$ git checkout fork
Switched to branch 'fork'
A       .gitignore
A       repository

bhuve@mittal MINGW64 /d/scm-lab2023 (fork)
$ git branch
* fork
  master

bhuve@mittal MINGW64 /d/scm-lab2023 (fork)
$ git status
On branch fork
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   .gitignore
        new file:   repository

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:   .gitignore
        modified:   repository (new commits)

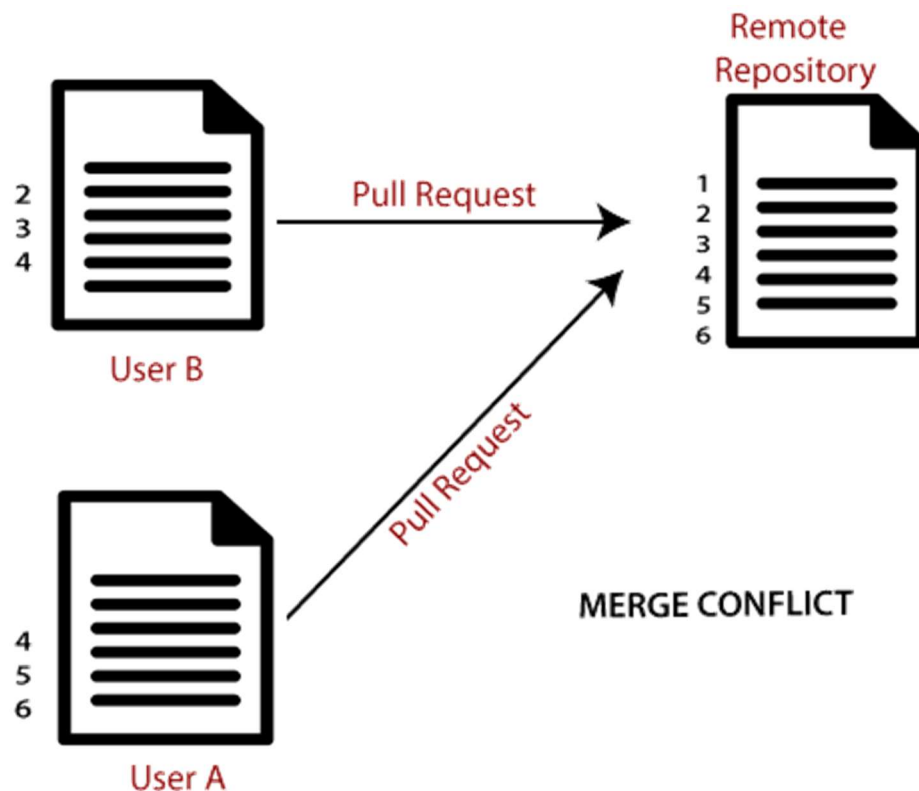
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        Moksh/

bhuve@mittal MINGW64 /d/scm-lab2023 (fork)
$ git add .
warning: adding embedded git repository: Moksh
hint: You've added another git repository inside your current repository.
hint: Clones of the outer repository will not contain the contents of
hint: the embedded repository and will not know how to obtain it.
hint: If you meant to add a submodule, use:
hint:
hint:   git submodule add <url> Moksh
hint:
hint: If you added this path by mistake, you can remove it from the
hint: index with:
hint:
hint:   git rm --cached Moksh
hint:
hint: See "git help submodule" for more information.

bhuve@mittal MINGW64 /d/scm-lab2023 (fork)
$ git status
On branch fork
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   .gitignore
        new file:   Moksh
        new file:   repository
```

Experiment 8. Merge and Resolve conflicts created due to own activity and collaborators activity.

Version control systems are all about managing contributions between multiple distributed authors (usually developers). Sometimes multiple developers may try to edit the same content. If Developer A tries to edit code that Developer B is editing a conflict may occur. reference for picture:



If you have a merge conflict on the command line, you cannot push your local changes to GitHub until you resolve the merge conflict locally on your computer.

To alleviate the occurrence of conflicts developers will work in separate isolated branches. If a merge conflict still arises between the compare branch and base branch in your pull request, you can view a list of the files with conflicting changes above the Merge pull request button. The Merge pull request button is deactivated until you've resolved all conflicts between the compare branch and base branch.

Step 1. Make changes in the master branch & commit them. Now checkout to another branch, make changes & commit them also.

```

MINGW64:/d/scm-lab2023

bhuve@mittal MINGW64 /d/scm-lab2023 (master)
$ touch f2.txt

bhuve@mittal MINGW64 /d/scm-lab2023 (master)
$ git add .

bhuve@mittal MINGW64 /d/scm-lab2023 (master)
$ git commit -m "merge conflicts 1"
[master e89bd49] merge conflicts 1
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 f2.txt

bhuve@mittal MINGW64 /d/scm-lab2023 (master)
$ git add .

bhuve@mittal MINGW64 /d/scm-lab2023 (master)
$ notepad f2.txt

bhuve@mittal MINGW64 /d/scm-lab2023 (master)
$ git commit -m "merge conflicts 2"
On branch 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:   f2.txt

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

bhuve@mittal MINGW64 /d/scm-lab2023 (master)
$ git add .

bhuve@mittal MINGW64 /d/scm-lab2023 (master)
$ git checkout conflicts
error: pathspec 'conflicts' did not match any file(s) known to git

bhuve@mittal MINGW64 /d/scm-lab2023 (master)
$ git branch conflicts

bhuve@mittal MINGW64 /d/scm-lab2023 (master)
$ git checkout conflicts
Switched to branch 'conflicts'
M       f2.txt

bhuve@mittal MINGW64 /d/scm-lab2023 (conflicts)
$ git commit -am "merge conflicts 3"
[conflicts da721cf] merge conflicts 3
1 file changed, 1 insertion(+)

bhuve@mittal MINGW64 /d/scm-lab2023 (conflicts)
$ git add .

```

Step 2. Now merge both the branches. It will give Conflicts error.


```
bhuve@mittal MINGW64 /d/scm-lab2023 (master)
$ git commit -m "merge conflicts 4"
[master aa7e168] merge conflicts 4
1 file changed, 1 insertion(+)

bhuve@mittal MINGW64 /d/scm-lab2023 (master)
$ git merge conflicts
Auto-merging f2.txt
CONFLICT (content): Merge conflict in f2.txt
Automatic merge failed; fix conflicts and then commit the result.
```

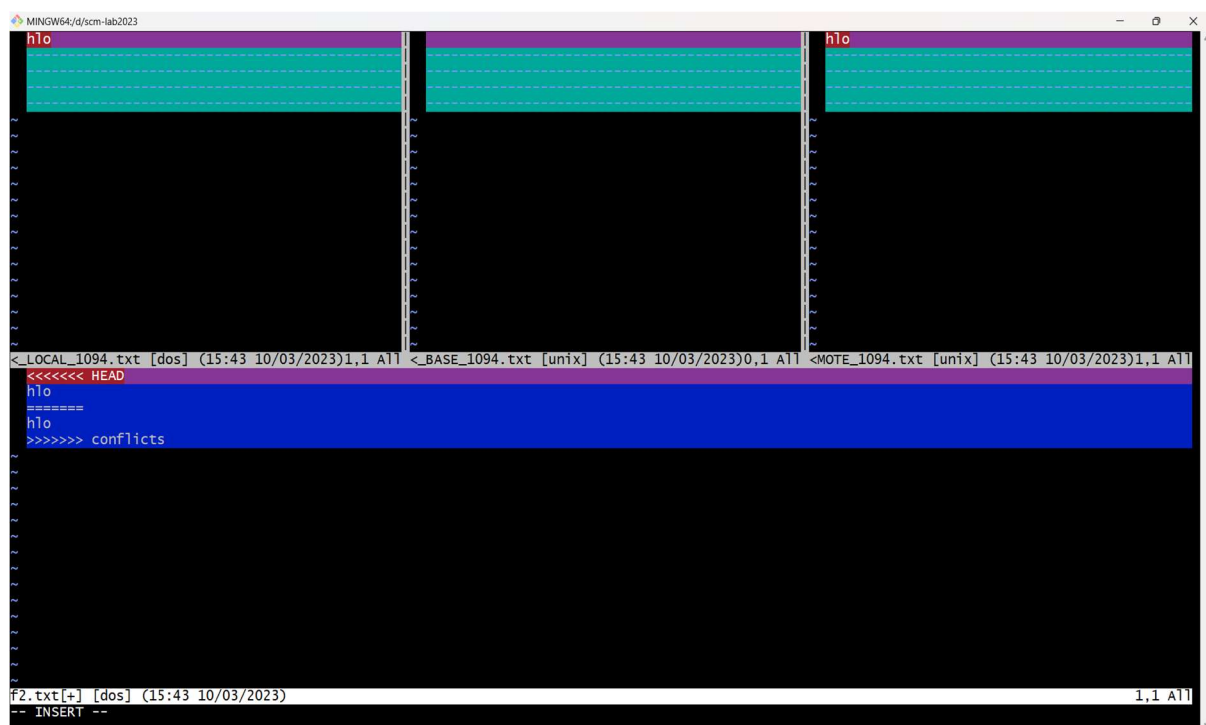
Step 3. Use **git mergetool** command to resolve the conflict. It runs the merge conflict resolution tools to resolve merge conflicts.

```
bhuve@mittal MINGW64 /d/scm-lab2023 (master|MERGING)
$ git mergetool

This message is displayed because 'merge.tool' is not configured.
See 'git mergetool --tool-help' or 'git help config' for more details.
'git mergetool' will now attempt to use one of the following tools:
opendiff kdiff3 tkdiff xxdiff meld tortoisemerge gvimdiff diffuse diffmerge ecmerge p4merge araxis bc codecompare smerge emerge vimdiff nvimdiff
Merging:
f2.txt

Normal merge conflict for 'f2.txt':
  {local}: modified file
  {remote}: modified file
Hit return to start merge resolution tool (vimdiff):
4 files to edit
```

Step 4. To insert, press **I**. Then type **:wq**. The merge conflict is solved and secondary branch is merged into the master branch.



```
hlo hlo hlo

<_LOCAL_1094.txt [dos] (15:43 10/03/2023)1,1 A11 <_BASE_1094.txt [unix] (15:43 10/03/2023)0,1 A11 <MOT_1094.txt [unix] (15:43 10/03/2023)1,1 A11
<<<<<< HEAD
hlo
=====
hlo
>>>>>> conflicts

f2.txt[*] [dos] (15:43 10/03/2023) 1,1 A11
-- INSERT --
```

Experiment 9. Reset and Revert

A reset is an operation that takes a specified commit and resets the "three trees" to match the state of the repository at that specified commit. A reset can be invoked in three different modes which correspond to the three trees. In reset, rest of the commits wash out after the mentioned commit. This is a limitation of reset command that we cannot have any random access.

A revert is an operation that takes a specified commit and creates a new commit which inverses the specified commit. git revert can only be run at a commit level scope and has no file level functionality.

These two features justify the Version- controlled feature of the git as we can rollback to any version at any time.

Reset: -

Step 1. Prepare a log of multiple commits to use reset command.

Step 2. Check git log.

```
bhuve@mittal MINGW64 /d/scm-lab2023 (master)
$ git log --oneline
76a64da (HEAD -> master) reset
762fb03 reset
86f7cef commit 2
60b9f8f commit 1
aa7e168 merge conflicts 4
e89bd49 merge conflicts 1
f9d5052 merge
314ad64 hi
```

Step 3. Pick any commit which we want the repository to rollback. Paste the checksum of the commit in the **git reset** command.

```

bhuve@mittal MINGW64 /d/scm-lab2023 (master)
$ git log --oneline
76a64da (HEAD -> master) reset 2
762fb03 reset
86f7cef commit 2
60b9f8f commit 1
aa7e168 merge conflicts 4
e89bd49 merge conflicts 1
f9d5052 merge
314ad64 hii

bhuve@mittal MINGW64 /d/scm-lab2023 (master)
$ git reset 86f7cef
Unstaged changes after reset:
M      f2.txt

bhuve@mittal MINGW64 /d/scm-lab2023 (master)
$ git log --oneline
86f7cef (HEAD -> master) commit 2
60b9f8f commit 1
aa7e168 merge conflicts 4
e89bd49 merge conflicts 1
f9d5052 merge
314ad64 hii

```

The head is now pointing the commit whose checksum we have provided that means the commit that followed vanished.

Revert: -

Step 1. Prepare a log of multiple commits to use reset command.

Step 2. Check git log.

```

bhuve@mittal MINGW64 /d/scm-lab2023 (master)
$ git log
commit 5292c8adec5c1109d5c4882d91a5fb6ce060d86e (HEAD -> master)
Author: Mittal854 <bhuveshmittal854@gmail.com>
Date:   Fri Mar 10 21:46:17 2023 +0530

    reset 2

commit 3ba218b41ec13edd85e3848b667affe8d1f5fdd9
Author: Mittal854 <bhuveshmittal854@gmail.com>
Date:   Fri Mar 10 21:44:53 2023 +0530

    reset

commit 86f7ceff63c8b28a64b55bdd829395f72d5a596f
Author: Mittal854 <bhuveshmittal854@gmail.com>
Date:   Fri Mar 10 21:36:57 2023 +0530

    commit 2

commit 60b9f8fa3de2c527f13f11450715d7b4f0f696ae
Author: Mittal854 <bhuveshmittal854@gmail.com>
Date:   Fri Mar 10 21:36:14 2023 +0530

    commit 1

commit aa7e16893a72757328cf51b205b09427de6f9ae0
Author: Mittal854 <bhuveshmittal854@gmail.com>
Date:   Fri Mar 10 15:42:11 2023 +0530

    merge conflicts 4

commit e89bd49d3c4a2f92e4bf1bc4c8b83c9d9e1d5a2e
Author: Mittal854 <bhuveshmittal854@gmail.com>
Date:   Fri Mar 10 15:36:14 2023 +0530

    merge conflicts 1

commit f9d5052d5ef35029d3813f15677115245cd39d4b
Author: Mittal854 <bhuveshmittal854@gmail.com>
Date:   Fri Mar 10 15:21:01 2023 +0530

    merge

commit 314ad64f79d6dcfc3ac7fc64ebc201019078ca0c
Author: Mittal854 <bhuveshmittal854@gmail.com>
Date:   Wed Feb 1 16:55:40 2023 +0530

    hii

```

Step 3. Pick up the commit which we want to revert. Paste the checksum in the **git revert** command.

```

bhuve@mittal MINGW64 /d/scm-lab2023 (master)
$ git revert 5292c8adec5c1109d5c4882d91a5fb6ce060d86e
[master 6a8a9be] Revert "reset 2"
 2 files changed, 1 insertion(+), 3 deletions(-)

```


Step 4. A window will appear. Enter the statement which we want to be displayed after reverting the commit.

Step 5. Check the git log. We will find that another commit is added without affecting the rest commits.

```
bhuve@mittal MINGW64 /d/scm-lab2023 (master)
$ git log
commit 6a8a9be686e262d537f1c23e6517815a1cfa1596 (HEAD -> master)
Author: Mittal854 <bhuveshmittal854@gmail.com>
Date:   Fri Mar 10 21:47:01 2023 +0530

    Revert "reset 2"

    This reverts commit 5292c8adec5c1109d5c4882d91a5fb6ce060d86e.

commit 5292c8adec5c1109d5c4882d91a5fb6ce060d86e
Author: Mittal854 <bhuveshmittal854@gmail.com>
Date:   Fri Mar 10 21:46:17 2023 +0530

    reset 2

commit 3ba218b41ec13edd85e3848b667affe8d1f5fdd9
Author: Mittal854 <bhuveshmittal854@gmail.com>
Date:   Fri Mar 10 21:44:53 2023 +0530

    reset

commit 86f7ceff63c8b28a64b55bdd829395f72d5a596f
Author: Mittal854 <bhuveshmittal854@gmail.com>
Date:   Fri Mar 10 21:36:57 2023 +0530

    commit 2

commit 60b9f8fa3de2c527f13f11450715d7b4f0f696ae
Author: Mittal854 <bhuveshmittal854@gmail.com>
Date:   Fri Mar 10 21:36:14 2023 +0530

    commit 1

commit aa7e16893a72757328cf51b205b09427de6f9ae0
Author: Mittal854 <bhuveshmittal854@gmail.com>
Date:   Fri Mar 10 15:42:11 2023 +0530

    merge conflicts 4

commit e89bd49d3c4a2f92e4bf1bc4c8b83c9d9e1d5a2e
Author: Mittal854 <bhuveshmittal854@gmail.com>
Date:   Fri Mar 10 15:36:14 2023 +0530

    merge conflicts 1

commit f9d5052d5ef35029d3813f15677115245cd39d4b
Author: Mittal854 <bhuveshmittal854@gmail.com>
Date:   Fri Mar 10 15:21:01 2023 +0530

    merge

commit 314ad64f79d6dcfc3ac7fc64ebc201019078ca0c
Author: Mittal854 <bhuveshmittal854@gmail.com>
Date:   Wed Feb 1 16:55:40 2023 +0530

    hii
```

The change associated to the reverted commit has disappeared.