Module - 6

Managing Source Code – Git and GitHub

Assignment

Date of submission - 2/01/2025

Submitted by - Chandra Sekhar

L1 -

Create Local git repository and demonstrate all git reset options and revert. Compare the differences.

Step - 1

Creating a local repository and initializing git init

```
MINGW64:/c/Users/Admin/assingment/repo

Admin@LAPTOP-C56V8DMF MINGW64 ~
$ mkdir assingment

Admin@LAPTOP-C56V8DMF MINGW64 ~
$ cd assingment/

Admin@LAPTOP-C56V8DMF MINGW64 ~/assingment
$ ls

Admin@LAPTOP-C56V8DMF MINGW64 ~/assingment
$ mkdir repo

Admin@LAPTOP-C56V8DMF MINGW64 ~/assingment
$ cd repo/

Admin@LAPTOP-C56V8DMF MINGW64 ~/assingment/repo
$ git init
Initialized empty Git repository in C:/Users/Admin/assingment/repo/.git/

Admin@LAPTOP-C56V8DMF MINGW64 ~/assingment/repo (master)
$
```

Step - 2

creating and adding and then Commiting 3 files in the repository

```
MINGW64:/c/Users/Admin/assingment/repo

Admin@LAPTOP-C56V8DMF MINGW64 ~/assingment/repo (master)
$ echo "hi" >> f1.txt

Admin@LAPTOP-C56V8DMF MINGW64 ~/assingment/repo (master)
$ git add .
warning: in the working copy of 'f1.txt', LF will be replaced by CRLF the next time Git touches it

Admin@LAPTOP-C56V8DMF MINGW64 ~/assingment/repo (master)
$ git commit -m "f1"
[master (root-commit) f73e287] f1
 1 file changed, 1 insertion(+)
 create mode 100644 f1.txt

Admin@LAPTOP-C56V8DMF MINGW64 ~/assingment/repo (master)
$ echo "bye" >> f2.txt

Admin@LAPTOP-C56V8DMF MINGW64 ~/assingment/repo (master)
$ git add .
warning: in the working copy of 'f2.txt', LF will be replaced by CRLF the next time Git touches it

Admin@LAPTOP-C56V8DMF MINGW64 ~/assingment/repo (master)
$ git commit -m "f2"
[master 89f4e2b] f2
 1 file changed, 1 insertion(+)
 create mode 100644 f2.txt
```

After commiting we can see all 3 files are present in working, staging area and are perfectly commited. We can see from “git status” command.

```
MINGW64:/c/Users/Admin/assingment/repo

Admin@LAPTOP-C56V8DMF MINGW64 ~/assingment/repo (master)
$ ls
f1.txt  f2.txt  f3.txt

Admin@LAPTOP-C56V8DMF MINGW64 ~/assingment/repo (master)
$ git log --oneline
e85cd43 (HEAD -> master) f3
89f4e2b f2
f73e287 f1

Admin@LAPTOP-C56V8DMF MINGW64 ~/assingment/repo (master)
$ git ls-files
f1.txt
f2.txt
f3.txt

Admin@LAPTOP-C56V8DMF MINGW64 ~/assingment/repo (master)
$ git status
On branch master
nothing to commit, working tree clean

Admin@LAPTOP-C56V8DMF MINGW64 ~/assingment/repo (master)
$ |
```

Step - 3

Using “git reset --soft <prev id>” command to remove commited from local repository

```
MINGW64:/c/Users/Admin/assingment/repo3

Admin@LAPTOP-C56V8DMF MINGW64 ~/assingment/repo3 (master)
$ ls
f2.txt

Admin@LAPTOP-C56V8DMF MINGW64 ~/assingment/repo3 (master)
$ git log --oneline
239cbb2 (HEAD -> master) yes Revert "f1"
db37e34 f2
d17342f f1

Admin@LAPTOP-C56V8DMF MINGW64 ~/assingment/repo3 (master)
$ git status
On branch master
nothing to commit, working tree clean

Admin@LAPTOP-C56V8DMF MINGW64 ~/assingment/repo3 (master)
$
```

After git soft command we can see that f3 file is removed from local repository but is present in staging and working directory

```
Admin@LAPTOP-C56V8DMF MINGW64 ~/assingment/repo (master)
$ git reset --soft 89f4e2b

Admin@LAPTOP-C56V8DMF MINGW64 ~/assingment/repo (master)
$ git log --oneline
89f4e2b (HEAD -> master) f2
f73e287 f1

Admin@LAPTOP-C56V8DMF MINGW64 ~/assingment/repo (master)
$ git ls-files
f1.txt
f2.txt
f3.txt

Admin@LAPTOP-C56V8DMF MINGW64 ~/assingment/repo (master)
$ ls
f1.txt  f2.txt  f3.txt

Admin@LAPTOP-C56V8DMF MINGW64 ~/assingment/repo (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   f3.txt
```

Step - 3

Now by using “git reset --mixed <prev id>” we remove file from local repository but is present at only working directory

```
MINGW64:/c/Users/Admin/assingment/repo

Admin@LAPTOP-C56V8DMF MINGW64 ~/assingment/repo (master)
$ git log --oneline
10023b2 (HEAD -> master) f3
000936d f2
f73e287 f1

Admin@LAPTOP-C56V8DMF MINGW64 ~/assingment/repo (master)
$ git reset --mixed 000936d

Admin@LAPTOP-C56V8DMF MINGW64 ~/assingment/repo (master)
$ git log --oneline
000936d (HEAD -> master) f2
f73e287 f1

Admin@LAPTOP-C56V8DMF MINGW64 ~/assingment/repo (master)
$ git ls-files
f1.txt
f2.txt

Admin@LAPTOP-C56V8DMF MINGW64 ~/assingment/repo (master)
$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        f3.txt

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

Step - 4

Now by using “git reset --hard <prev id>” we remove file from local repository,working and staging area completely

```
MINGW64:/c/Users/Admin/assingment/repo

Admin@LAPTOP-C56V8DMF MINGW64 ~/assingment/repo (master)
$ git log --oneline
7a893b5 (HEAD -> master) f3
000936d f2
f73e287 f1

Admin@LAPTOP-C56V8DMF MINGW64 ~/assingment/repo (master)
$ git reset --hard 000936d
HEAD is now at 000936d f2

Admin@LAPTOP-C56V8DMF MINGW64 ~/assingment/repo (master)
$ ls
f1.txt  f2.txt

Admin@LAPTOP-C56V8DMF MINGW64 ~/assingment/repo (master)
$ git log --oneline
000936d (HEAD -> master) f2
f73e287 f1

Admin@LAPTOP-C56V8DMF MINGW64 ~/assingment/repo (master)
$ git ls-files
f1.txt
f2.txt

Admin@LAPTOP-C56V8DMF MINGW64 ~/assingment/repo (master)
$ git status
On branch master
nothing to commit, working tree clean

Admin@LAPTOP-C56V8DMF MINGW64 ~/assingment/repo (master)
$
```

Step - 5

Now by using “git revert <specific id>” we remove file from local repository,working and staging area completely but there a difference from git hard

```
MINGW64:/c/Users/Admin/assingment/repo3

Admin@LAPTOP-C56V8DMF MINGW64 ~/assingment/repo3 (master)
$ ls
f1.txt  f2.txt

Admin@LAPTOP-C56V8DMF MINGW64 ~/assingment/repo3 (master)
$ git log --oneline
db37e34 (HEAD -> master) f2
d17342f f1

Admin@LAPTOP-C56V8DMF MINGW64 ~/assingment/repo3 (master)
$ git ls-files
f1.txt
f2.txt

Admin@LAPTOP-C56V8DMF MINGW64 ~/assingment/repo3 (master)
$ git revert d17342f
```

```
MINGW64:/c/Users/Admin/assingment/repo3
Revert "f1"

This reverts commit d17342f003ce6ff754fc4cbd32ad0cf7256de3ca.

# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
#
# On branch master
# Changes to be committed:
#       deleted:    f1.txt
#
```

```
[master 239cbb2] yes Revert "f1"
 1 file changed, 1 deletion(-)
 delete mode 100644 f1.txt

Admin@LAPTOP-C56V8DMF MINGW64 ~/assingment/repo3 (master)
$ |
```

```
MINGW64:/c/Users/Admin/assingment/repo3

Admin@LAPTOP-C56V8DMF MINGW64 ~/assingment/repo3 (master)
$ ls
f2.txt

Admin@LAPTOP-C56V8DMF MINGW64 ~/assingment/repo3 (master)
$ git log --oneline
239cbb2 (HEAD -> master) yes Revert "f1"
db37e34 f2
d17342f f1

Admin@LAPTOP-C56V8DMF MINGW64 ~/assingment/repo3 (master)
$ git status
On branch master
nothing to commit, working tree clean

Admin@LAPTOP-C56V8DMF MINGW64 ~/assingment/repo3 (master)
$ |
```

Differences

The main difference is that in git reset there is no commit history of any but in git revert their is commit history.

L2-

Create Local git repository and demonstrate git merge and Merge Conflicts with the steps to resolve merge conflicts

Step - 1

create a merge directory and initilize git in it

```
MINGW64:/d/merge/repo1

Admin@LAPTOP-C56V8DMF MINGW64 ~
$ cd d:

Admin@LAPTOP-C56V8DMF MINGW64 /d
$ mkdir merge

Admin@LAPTOP-C56V8DMF MINGW64 /d
$ cd merge

Admin@LAPTOP-C56V8DMF MINGW64 /d/merge
$ mkdir repo1

Admin@LAPTOP-C56V8DMF MINGW64 /d/merge
$ cd repo1

Admin@LAPTOP-C56V8DMF MINGW64 /d/merge/repo1
$ git init
Initialized empty Git repository in D:/merge/repo1/.git/

Admin@LAPTOP-C56V8DMF MINGW64 /d/merge/repo1 (master)
$ |
```

Now commiting 3 files to the repo and creating feature1 and feature2 branches.

```
MINGW64:/d/merge/repo1

Admin@LAPTOP-C56V8DMF MINGW64 /d/merge/repo1 (master)
$ echo "dsef" >> f1.txt

Admin@LAPTOP-C56V8DMF MINGW64 /d/merge/repo1 (master)
$ git add .
warning: in the working copy of 'f1.txt', LF will be replaced by CRLF the next time Git touches it

Admin@LAPTOP-C56V8DMF MINGW64 /d/merge/repo1 (master)
$ git commit -m "cm1"
[master (root-commit) d95f98b] cm1
 1 file changed, 1 insertion(+)
 create mode 100644 f1.txt

Admin@LAPTOP-C56V8DMF MINGW64 /d/merge/repo1 (master)
$ echo "sdcs" >> f2.txt

Admin@LAPTOP-C56V8DMF MINGW64 /d/merge/repo1 (master)
$ git add .
warning: in the working copy of 'f2.txt', LF will be replaced by CRLF the next time Git touches it

Admin@LAPTOP-C56V8DMF MINGW64 /d/merge/repo1 (master)
$ git commit -m "cm2"
[master e378980] cm2
 1 file changed, 1 insertion(+)
 create mode 100644 f2.txt

Admin@LAPTOP-C56V8DMF MINGW64 /d/merge/repo1 (master)
$ echo "eda" >> f3.txt

Admin@LAPTOP-C56V8DMF MINGW64 /d/merge/repo1 (master)
$ git add .
warning: in the working copy of 'f3.txt', LF will be replaced by CRLF the next time Git touches it

Admin@LAPTOP-C56V8DMF MINGW64 /d/merge/repo1 (master)
$ git commit -m "cm3"
[master 4a94f6e] cm3
 1 file changed, 1 insertion(+)
 create mode 100644 f3.txt

Admin@LAPTOP-C56V8DMF MINGW64 /d/merge/repo1 (master)
$
```

```
MINGW64:/d/merge/repo1

Admin@LAPTOP-C56V8DMF MINGW64 /d/merge/repo1 (master)
$ ls
f1.txt  f2.txt  f3.txt

Admin@LAPTOP-C56V8DMF MINGW64 /d/merge/repo1 (master)
$ git log --oneline
4a94f6e (HEAD -> master) cm3
e378980 cm2
d95f98b cm1

Admin@LAPTOP-C56V8DMF MINGW64 /d/merge/repo1 (master)
$ |
```

```
MINGW64:/d/merge/repo1

Admin@LAPTOP-C56V8DMF MINGW64 /d/merge/repo1 (master)
$ git branch feature1

Admin@LAPTOP-C56V8DMF MINGW64 /d/merge/repo1 (master)
$ git branch feature2

Admin@LAPTOP-C56V8DMF MINGW64 /d/merge/repo1 (master)
$ git log --oneline
4a94f6e (HEAD -> master, feature2, feature1) cm3
e378980 cm2
d95f98b cm1

Admin@LAPTOP-C56V8DMF MINGW64 /d/merge/repo1 (master)
$
```

```
MINGW64:/d/merge/repo1

dmin@LAPTOP-C56V8DMF MINGW64 /d/merge/repo1 (master)
 git branch
  feature1
  feature2
* master

dmin@LAPTOP-C56V8DMF MINGW64 /d/merge/repo1 (master)
```

Step - 2

now switching to feature 1 branch and commiting few changes in file1.txt then merging the feature branch in master branch.

```
MINGW64:/d/merge/repo1

Admin@LAPTOP-C56V8DMF MINGW64 /d/merge/repo1 (feature1)
$ ls
f1.txt  f2.txt  f3.txt

Admin@LAPTOP-C56V8DMF MINGW64 /d/merge/repo1 (feature1)
$ git log --oneline
4a94f6e (HEAD -> feature1, master, feature2) cm3
e378980 cm2
d95f98b cm1

Admin@LAPTOP-C56V8DMF MINGW64 /d/merge/repo1 (feature1)
$ echo "record from f1" >> f1.txt

Admin@LAPTOP-C56V8DMF MINGW64 /d/merge/repo1 (feature1)
$ git add .
warning: in the working copy of 'f1.txt', LF will be replaced by CRLF the next time Git touches it

Admin@LAPTOP-C56V8DMF MINGW64 /d/merge/repo1 (feature1)
$ git commit -m "fcm1"
[feature1 84957cc] fcm1
 1 file changed, 1 insertion(+)

Admin@LAPTOP-C56V8DMF MINGW64 /d/merge/repo1 (feature1)
$ |
```

```
Admin@LAPTOP-C56V8DMF MINGW64 /d/merge/repo1 (feature1)
$ git log --oneline
84957cc (HEAD -> feature1) fcm1
4a94f6e (master, feature2) cm3
e378980 cm2
d95f98b cm1

Admin@LAPTOP-C56V8DMF MINGW64 /d/merge/repo1 (feature1)
$ git switch master
Switched to branch 'master'

Admin@LAPTOP-C56V8DMF MINGW64 /d/merge/repo1 (master)
$ git merge feature1
Updating 4a94f6e..84957cc
Fast-forward
 f1.txt | 1 +
 1 file changed, 1 insertion(+)

Admin@LAPTOP-C56V8DMF MINGW64 /d/merge/repo1 (master)
$ git log --oneline
84957cc (HEAD -> master, feature1) fcm1
4a94f6e (feature2) cm3
e378980 cm2
d95f98b cm1

Admin@LAPTOP-C56V8DMF MINGW64 /d/merge/repo1 (master)
$
```

Step - 3

now switching to feature 2 branch and commiting few changes in file1.txt then merging the feature branch in master branch. while merging conflict occurs because we are updating the same file at the same time.

```
MINGW64:/d/merge/repo1

Admin@LAPTOP-C56V8DMF MINGW64 /d/merge/repo1 (master)
$ git switch feature2
Switched to branch 'feature2'

Admin@LAPTOP-C56V8DMF MINGW64 /d/merge/repo1 (feature2)
$ ls
f1.txt  f2.txt  f3.txt

Admin@LAPTOP-C56V8DMF MINGW64 /d/merge/repo1 (feature2)
$ git log --oneline
4a94f6e (HEAD -> feature2) cm3
e378980 cm2
d95f98b cm1

Admin@LAPTOP-C56V8DMF MINGW64 /d/merge/repo1 (feature2)
$ echo "record from f2" >> f1.txt

Admin@LAPTOP-C56V8DMF MINGW64 /d/merge/repo1 (feature2)
$ git add .
warning: in the working copy of 'f1.txt', LF will be replaced by CRLF the next time Git touches it

Admin@LAPTOP-C56V8DMF MINGW64 /d/merge/repo1 (feature2)
$ git commit -m "f1cm2"
[feature2 6d5fdf1] f1cm2
 1 file changed, 1 insertion(+)

Admin@LAPTOP-C56V8DMF MINGW64 /d/merge/repo1 (feature2)
$ git log --oneline
6d5fdf1 (HEAD -> feature2) f1cm2
4a94f6e cm3
e378980 cm2
d95f98b cm1

Admin@LAPTOP-C56V8DMF MINGW64 /d/merge/repo1 (feature2)
$ |
```

```
MINGW64:/d/merge/repo1

Admin@LAPTOP-C56V8DMF MINGW64 /d/merge/repo1 (feature2)
$ git switch master
Switched to branch 'master'

Admin@LAPTOP-C56V8DMF MINGW64 /d/merge/repo1 (master)
$ git merge feature2
Auto-merging f1.txt
CONFLICT (content): Merge conflict in f1.txt
Automatic merge failed; fix conflicts and then commit the result.

Admin@LAPTOP-C56V8DMF MINGW64 /d/merge/repo1 (master|MERGING)
$ git status
On branch master
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:   f1.txt

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

Admin@LAPTOP-C56V8DMF MINGW64 /d/merge/repo1 (master|MERGING)
$ |
```

Step - 4

1. Identify the file(s) causing the Merge Conflict
2. Open and review the file content
3. Upon review, decide which record should be retained or deleted from that file
4. Open the File in edit mode, and delete the header and footer lines, and update the file with required records and save it.
5. perform git add and commit to the target branch

```
MINGW64:/d/merge/repo1

Admin@LAPTOP-C56V8DMF MINGW64 /d/merge/repo1 (master|MERGING)
$ git status
On branch master
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:   f1.txt

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

Admin@LAPTOP-C56V8DMF MINGW64 /d/merge/repo1 (master|MERGING)
$ cat f1.txt
dsef
<<<<<<< HEAD
record from f1
=======
record from f2
>>>>>>> feature2

Admin@LAPTOP-C56V8DMF MINGW64 /d/merge/repo1 (master|MERGING)
$ vi f1.txt
```

```
MINGW64:/d/merge/repo1
dsef
<<<<<<< HEAD
record from f1
=======
record from f2
>>>>>>> feature2
~
~
~
```

MINGW64:/d/merge/repo1

```
dsef
record from f1
record from f2
~
```

MINGW64:/d/merge/repo1

```
Admin@LAPTOP-C56V8DMF MINGW64 /d/merge/repo1 (master|MERGING)
$ git status
On branch master
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:   f1.txt

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

Admin@LAPTOP-C56V8DMF MINGW64 /d/merge/repo1 (master|MERGING)
$ cat f1.txt
dsef
<<<<<<< HEAD
record from f1
=======
record from f2
>>>>>>> feature2

Admin@LAPTOP-C56V8DMF MINGW64 /d/merge/repo1 (master|MERGING)
$ vi f1.txt

Admin@LAPTOP-C56V8DMF MINGW64 /d/merge/repo1 (master|MERGING)
$ git add .

Admin@LAPTOP-C56V8DMF MINGW64 /d/merge/repo1 (master|MERGING)
$ git commit -m "mcm"
[master 5e0b86c] mcm

Admin@LAPTOP-C56V8DMF MINGW64 /d/merge/repo1 (master)
$ git status
On branch master
nothing to commit, working tree clean
```

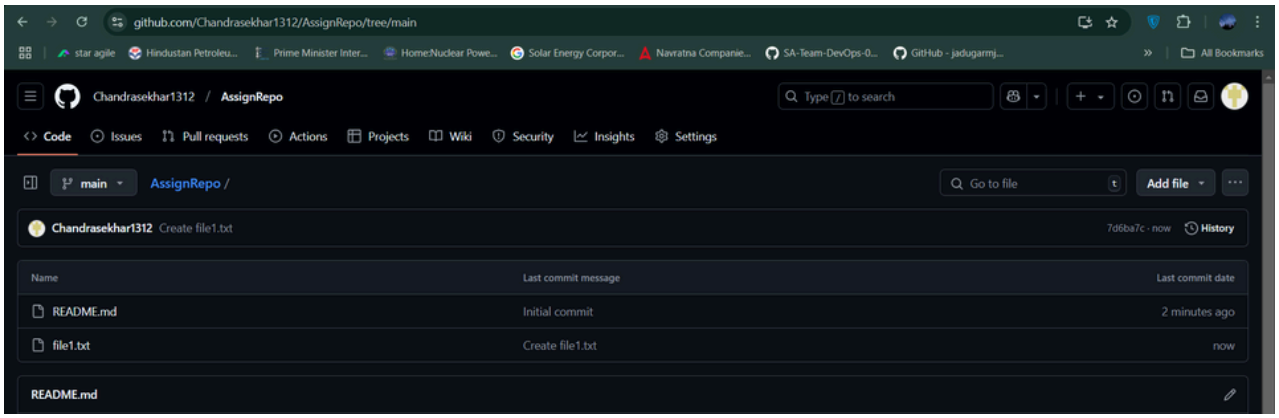
L3-

Using Local and Remote git repositories demonstrate git pull and git fetch. Compare the differences

Step - 3

First creating a local repository and cloning the remote repository in it

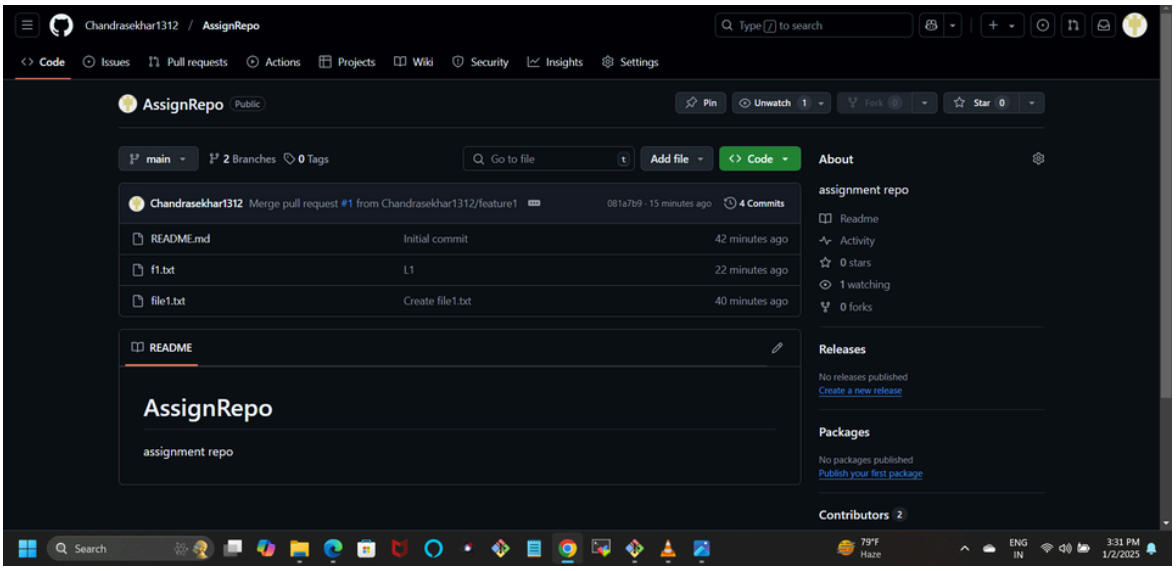
```
MINGW64:/d/localrepo

Admin@LAPTOP-C56V8DMF MINGW64 /d
$ mkdir localrepo

Admin@LAPTOP-C56V8DMF MINGW64 /d
$ cd localrepo/

Admin@LAPTOP-C56V8DMF MINGW64 /d/localrepo
$ git init
Initialized empty Git repository in D:/localrepo/.git/

Admin@LAPTOP-C56V8DMF MINGW64 /d/localrepo (master)
$
```



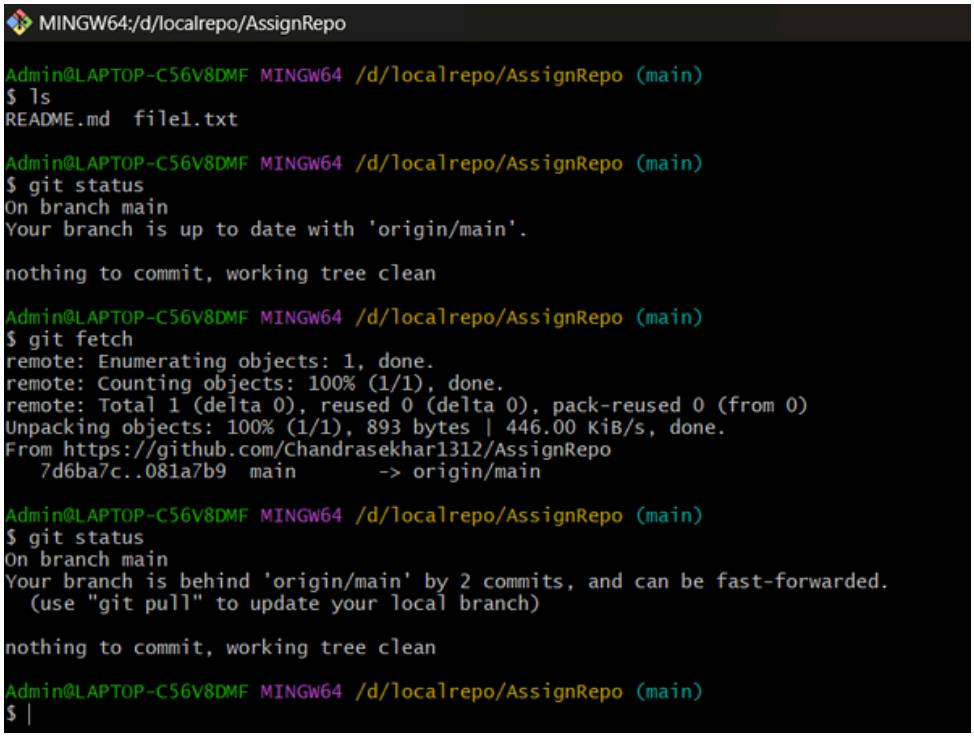

```
MINGW64:/d/localrepo/AssignRepo

Admin@LAPTOP-C56V8DMF MINGW64 /d/localrepo (master)
$ git clone https://github.com/Chandrasekhar1312/AssignRepo.git
Cloning into 'AssignRepo'...
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 (from 0)
Receiving objects: 100% (6/6), done.

Admin@LAPTOP-C56V8DMF MINGW64 /d/localrepo (master)
$ ls
AssignRepo/

Admin@LAPTOP-C56V8DMF MINGW64 /d/localrepo (master)
$ cd AssignRepo/

Admin@LAPTOP-C56V8DMF MINGW64 /d/localrepo/AssignRepo (main)
$ ls
README.md  file1.txt

Admin@LAPTOP-C56V8DMF MINGW64 /d/localrepo/AssignRepo (main)
$ |
```

Step - 4

Here we made a new commmit in remote repository now we need to update it in local repository by using git fetch and git pull command



we can see that local repo has only one file after “git fetch” command we can see that our localbranch is 2 commit behind then we use “git pull” command to update the local repository and working directory.



```
MINGW64:/d/localrepo/AssignRepo

Admin@LAPTOP-C56V8DMF MINGW64 /d/localrepo/AssignRepo (main)
$ ls
README.md  file1.txt

Admin@LAPTOP-C56V8DMF MINGW64 /d/localrepo/AssignRepo (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean

Admin@LAPTOP-C56V8DMF MINGW64 /d/localrepo/AssignRepo (main)
$ git fetch
remote: Enumerating objects: 1, done.
remote: Counting objects: 100% (1/1), done.
remote: Total 1 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Unpacking objects: 100% (1/1), 893 bytes | 446.00 KiB/s, done.
From https://github.com/Chandrasekhar1312/AssignRepo
   7d6ba7c..081a7b9  main       -> origin/main

Admin@LAPTOP-C56V8DMF MINGW64 /d/localrepo/AssignRepo (main)
$ git status
On branch main
Your branch is behind 'origin/main' by 2 commits, and can be fast-forwarded.
  (use "git pull" to update your local branch)

nothing to commit, working tree clean

Admin@LAPTOP-C56V8DMF MINGW64 /d/localrepo/AssignRepo (main)
$ |
```

The diffrence between pull and fetch is that, Fetch command only updates the local repository about new commits. and Pull command updates both local repository and working directory.

```
MINGW64:/d/localrepo/AssignRepo

Admin@LAPTOP-C56V8DMF MINGW64 /d/localrepo/AssignRepo (main)
$ git pull
Updating 7d6ba7c..081a7b9
Fast-forward
 f1.txt | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 f1.txt

Admin@LAPTOP-C56V8DMF MINGW64 /d/localrepo/AssignRepo (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean

Admin@LAPTOP-C56V8DMF MINGW64 /d/localrepo/AssignRepo (main)
$ ls
README.md  f1.txt  file1.txt

Admin@LAPTOP-C56V8DMF MINGW64 /d/localrepo/AssignRepo (main)
$
```

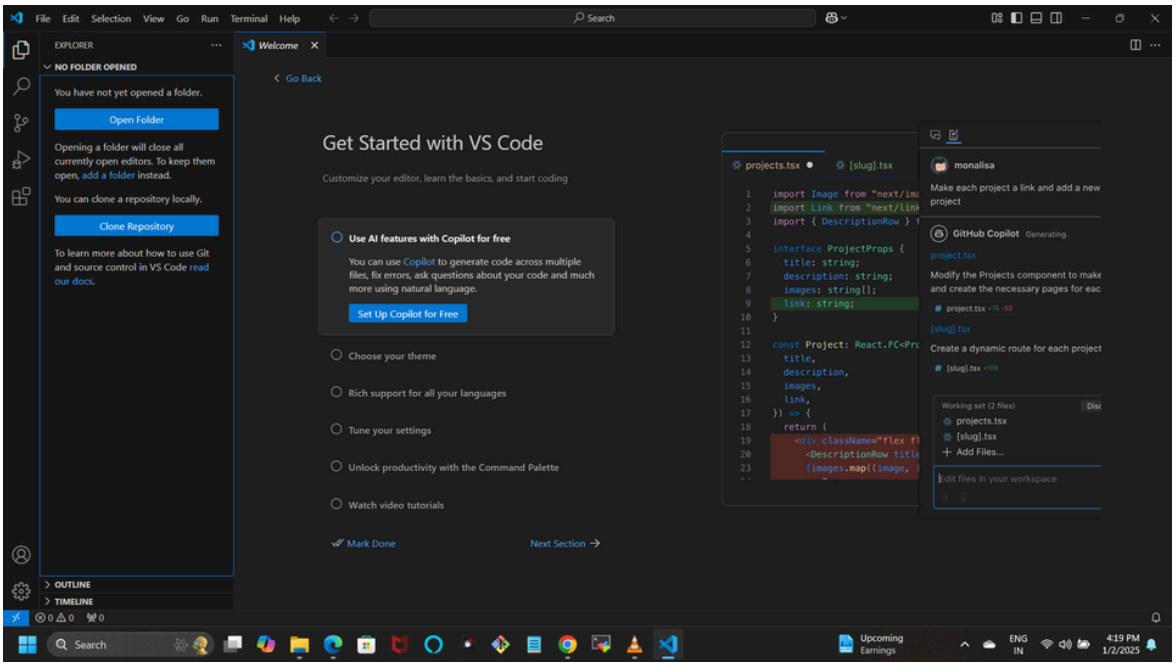
After the pull command we can see that the local repository is up to date with remote repository.

L4 -

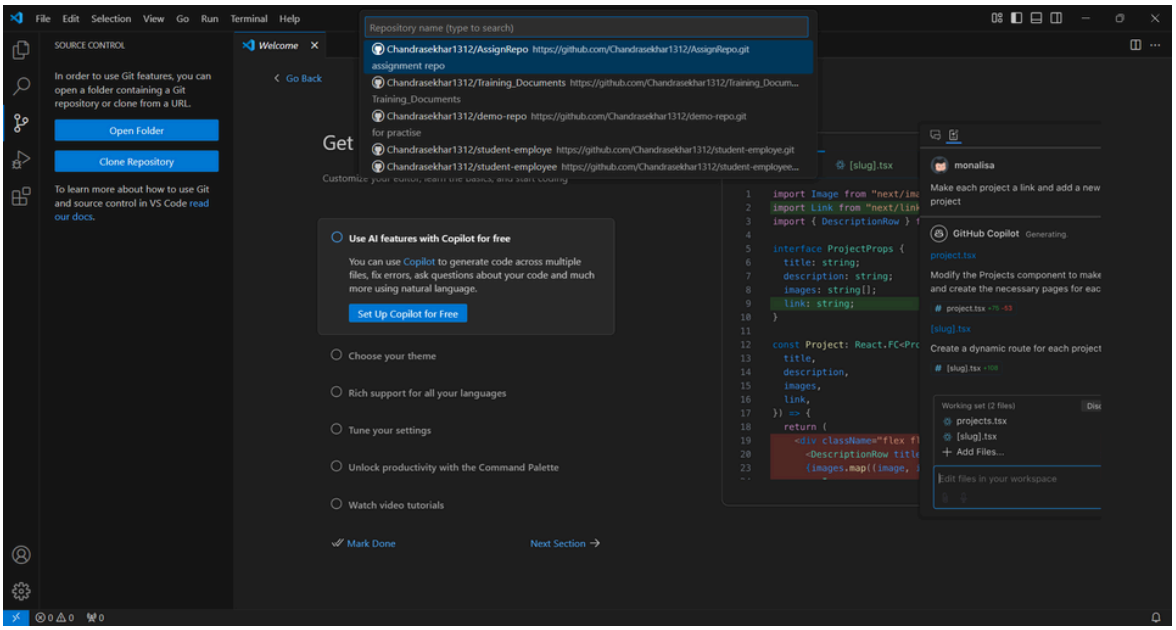
Clone GitHub repository using Visual Studio Code IDE

Step - 1

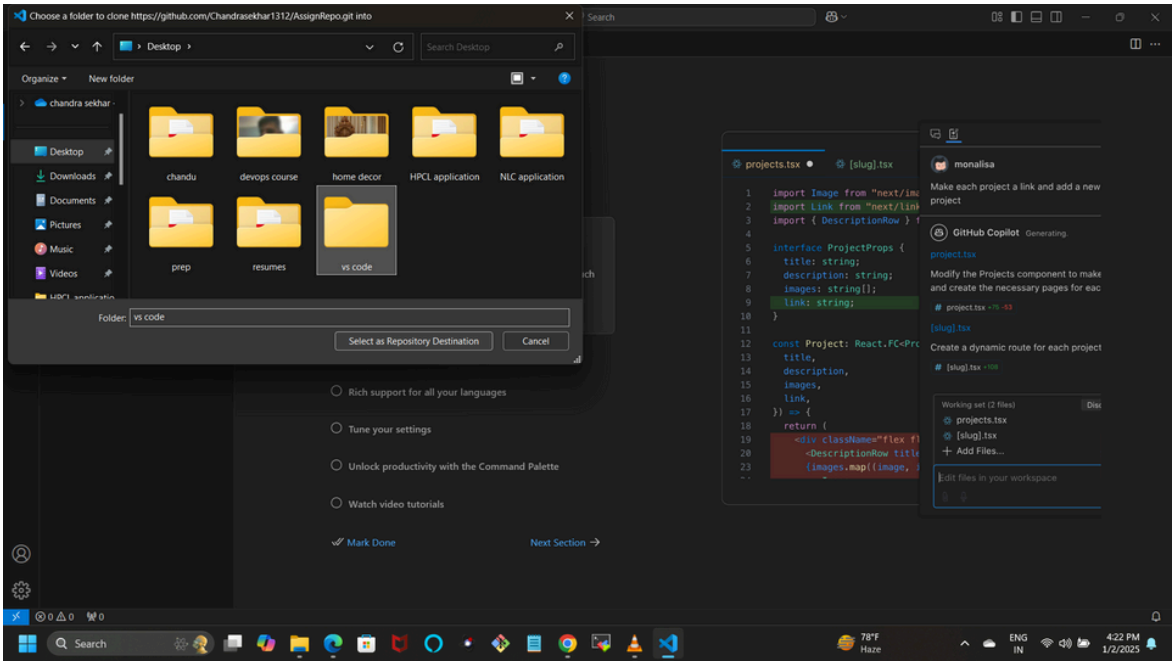
After opening vscode navigate to file explorer, here we can see clone repository option. click on it



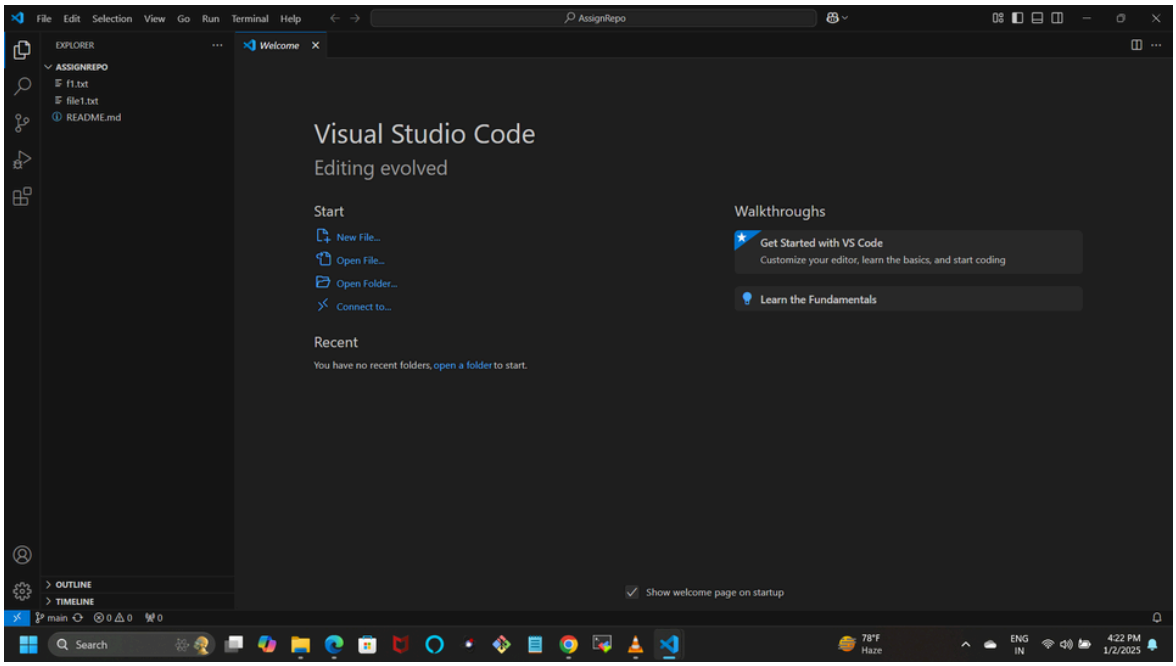
In the search bar we need to paste our repository url but here my account is already linked so its diectly showing to choose which repository. Select the one we want.



After that we need to choose a folder to save the repository



After selecting we have our repository in vscode.

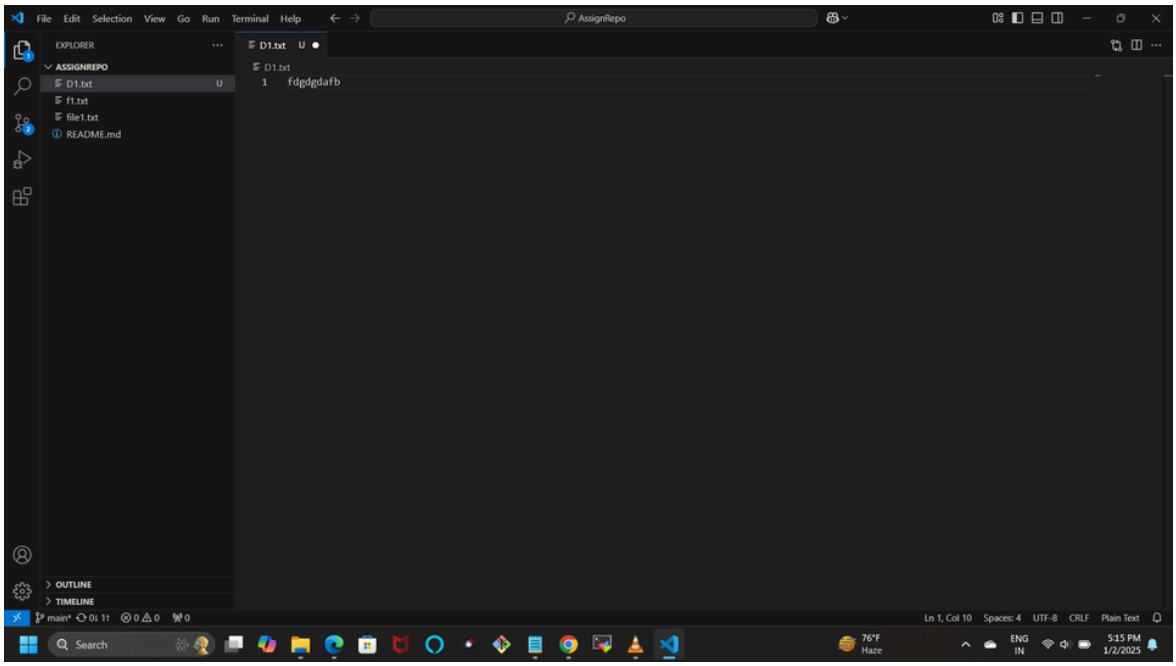


L5 -

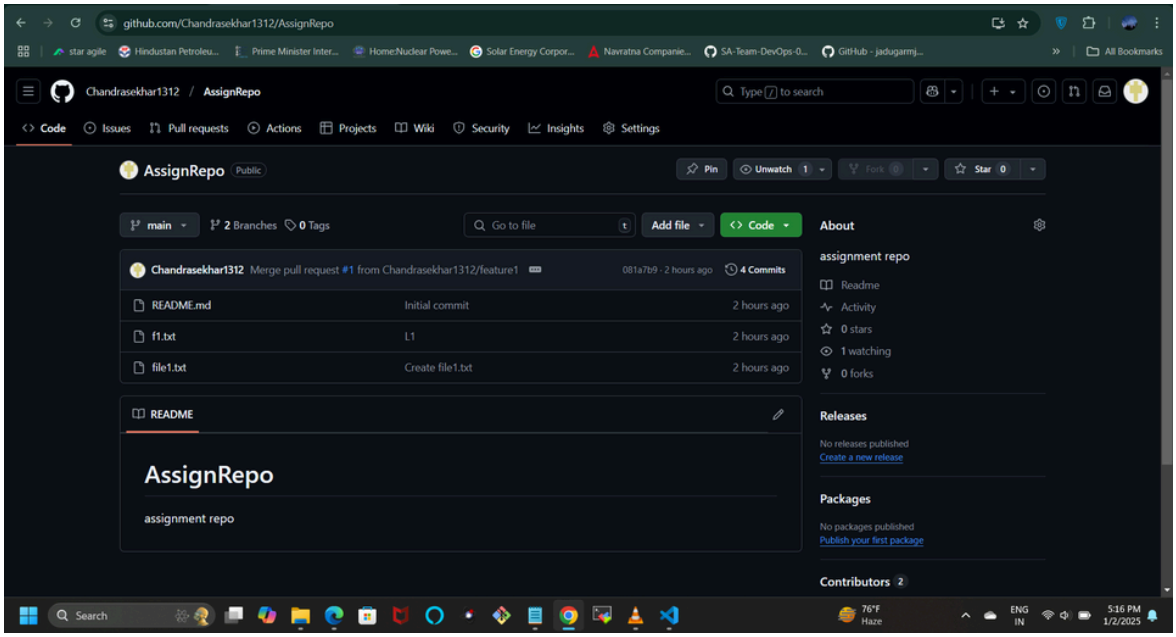
Push the incremental changes to GitHub Repository through Visual Studio Code IDE

Step - 1

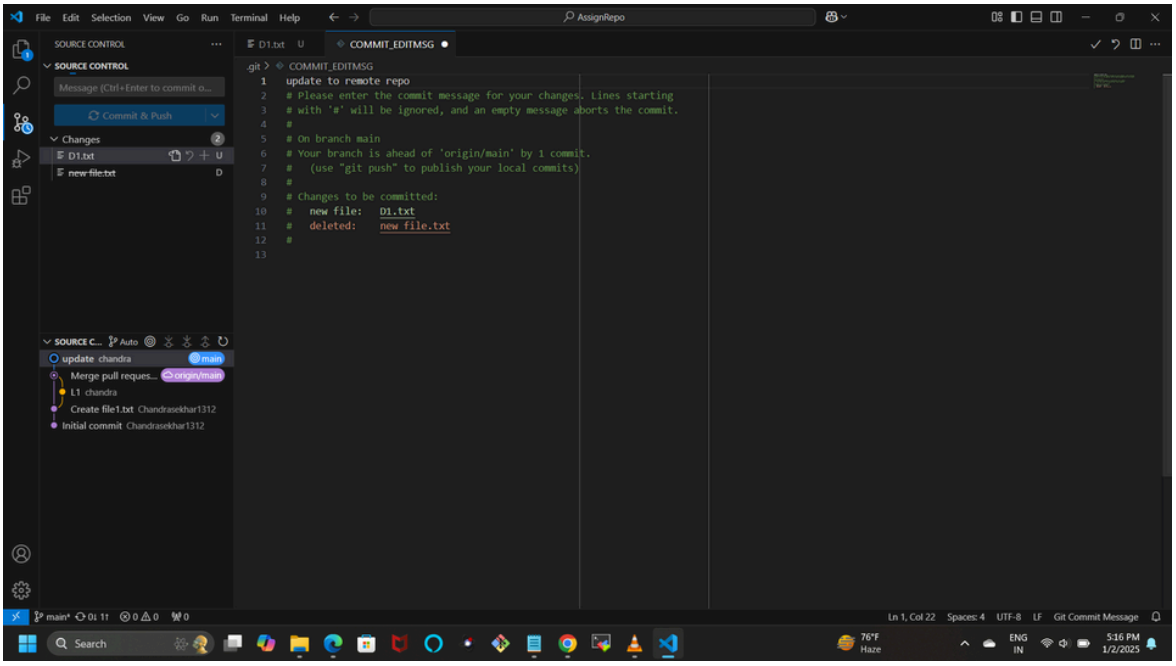
In file explorer we can see add new file option, click on that and add a new file.



we can see the remote repository has 2 files before adding any new file to it froom vscode



Now navigationg to source control option we can see commit and push option click on that we need to type commit mesage that it.



After that we can see that remote repository has 3 files after pushing new commits from vscode repository to remote repository