Subject name: **Source code management**

Subject code: **CS181**

Cluster: **Beta**

Department: **DCSE**

Submitted by:Adarsh Raj
Submitted to: Dr. Ankit Bansal
2110990075
G8

Department of Computer Science & Engineering

Chitkara University Institute of Engineering and Technology, Punjab

Jan- June
(2021-22)

CHITKARA UNIVERSITY

Institute/School Name	**Chitkara University Institute of Engineering and Technology**		
Department Name	**Department of Computer Science & Engineering**		
Programme Name	**Bachelor of Engineering (B.E.), Computer Science & Engineering**		
Course Name	**Source Code Management**	Session	**2021-22**
Course Code	**CS181**	Semester/Batch	**2nd/2021**
Vertical Name	**Beta**	Group No	G8
Course Coordinator	**Dr. Neeraj Singla**		
Faculty Name	**Dr. Ankit Bansal**		

Submission

Name: Japleen kaur
Signature: Japleen kaur

Date: 23-5-2022

CHITKARA UNIVERSITY

Table of Content

1. Version control with Git

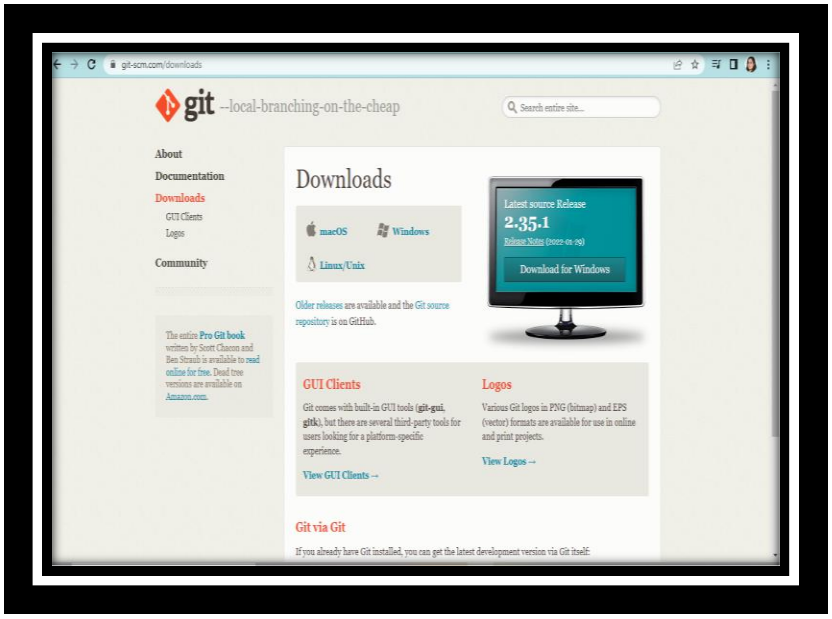
Step 1 - Installing Git:

Before we start using Git, we have to make it available on your computer. Even if it's already Installed , it's probably a good idea to update to the latest version. We can either install it as a package or via another installer, or download the source code and compile it yourself.

Step 2 - Installing on windows:

There are also a few ways to install Git on Windows. The most official build is available for download on the Git website. Just go to https://git-scm.com/download/win and the download will start automatically. Note that this is a project called Git for Windows, which is separate from Git itself; for more information on it, go to https://gitforwindows.org.

1.



The entire Pro Git book written by Scott Chacon and Ben Straub is available to read online for free. Dead tree versions are available on Amazon.com.

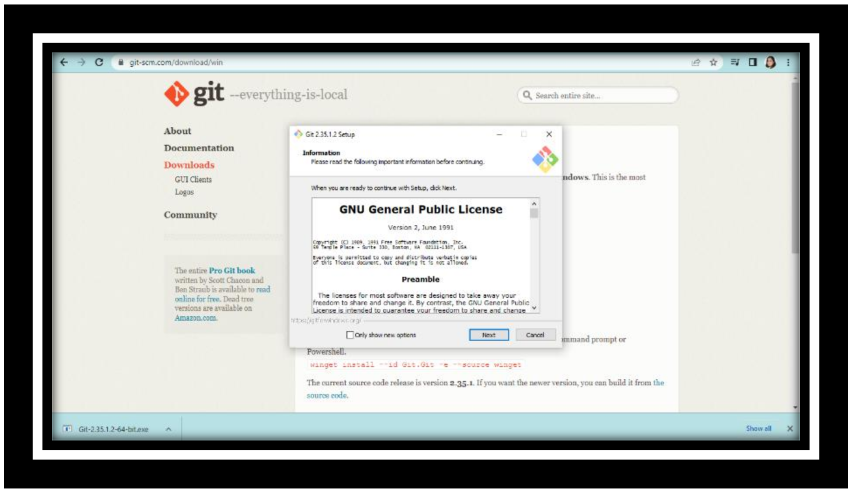
Older releases are available and the Git source repository is on GitHub.

Git comes with built-in GUI tools (git-gui, gitk), but there are several third-party tools for users looking for a platform-specific experience.

Various Git logos in PNG (bitmap) and EPS (vector) formats are available for use in online and print projects.

If you already have Git installed, you can get the latest development version via Git itself:

2.



The entire Pro Git book written by Scott Chacon and Ben Straub is available to read online for free. Dead tree versions are available on Amazon.com.

Information

Please read the following important information before continuing.

When you are ready to continue with Setup, click Next.

GNU General Public License

Version 2, June 1991

Copyright (C) 1989, 1991 Free Software Foundation, Inc.
59 Temple Place - Suite 330, Boston, MA 02111-1307, USA

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change

https://gitforwindows.org/

[ ] Only show new options

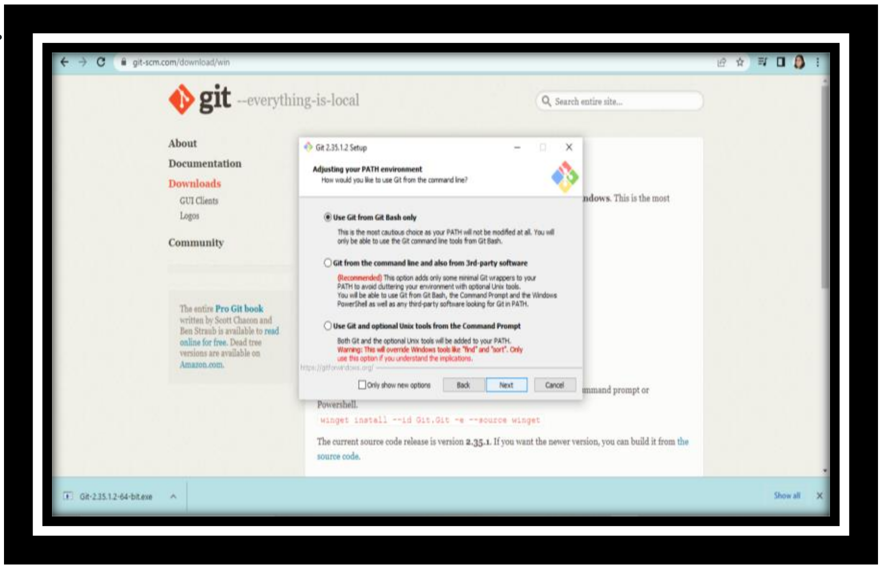
...ndows. This is the most

...mmand prompt or Powershell.

```
winget install --id Git.Git -e --source winget
```

The current source code release is version **2.35.1**. If you want the newer version, you can build it from the source code.

3.



Git 2.35.1.2 Setup

Adjusting your PATH environment
How would you like to use Git from the command line?

[✓ Use Git from Git Bash only]
This is the most cautious choice as your PATH will not be modified at all. You will only be able to use the Git command line tools from Git Bash.

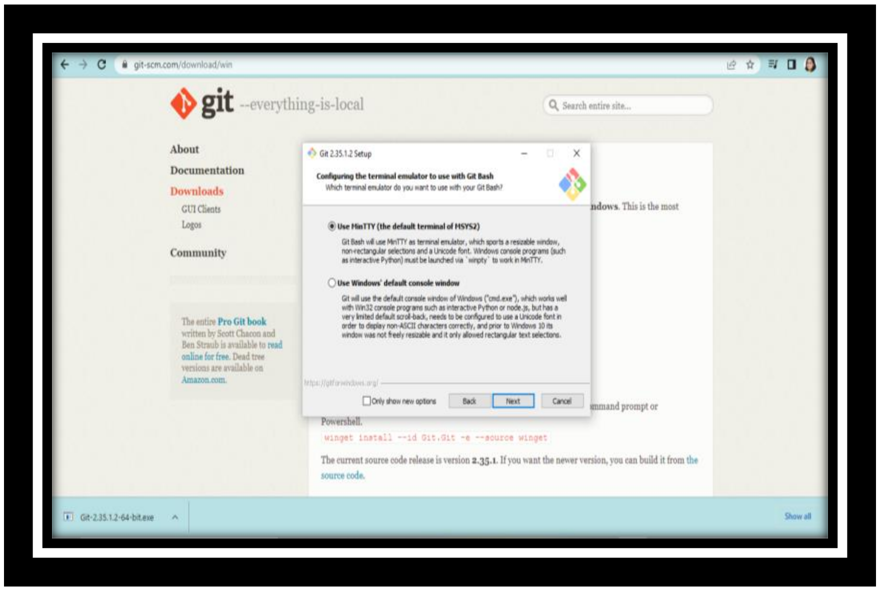
Git from the command line and also from 3rd-party software
(Recommended) This option adds only some minimal Git wrappers to your PATH to avoid cluttering your environment with optional Unix tools. You will be able to use Git from Git Bash, the Command Prompt and the Windows PowerShell as well as any third-party software looking for Git in PATH.

Use Git and optional Unix tools from the Command Prompt
Both Git and the optional Unix tools will be added to your PATH.
Warning: This will override Windows tools like "find" and "sort". Only use this option if you understand the implications.

```
winget install --id Git.Git -e --source winget
```

The current source code release is version **2.35.1**. If you want the newer version, you can build it from the source code.

4.



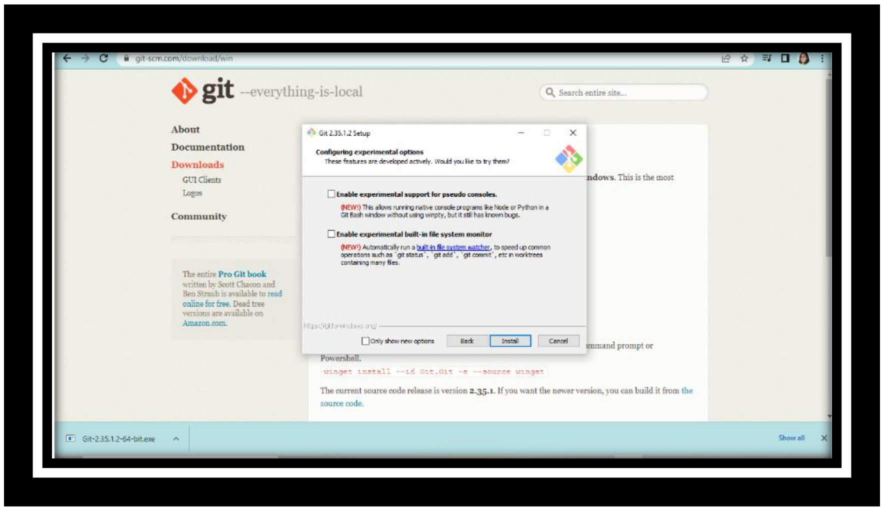
Git 2.35.1.2 Setup

Configuring the terminal emulator to use with Git Bash
Which terminal emulator do you want to use with your Git Bash?

[✓ Use MinTTY (the default terminal of MSYS2)]
Git Bash will use MinTTY as terminal emulator, which sports a resizable window, non-rectangular selections and a Unicode font. Windows console programs (such as interactive Python) must be launched via `winpty` to work in MinTTY.

Use Windows' default console window
Git will use the default console window of Windows ("cmd.exe"), which works well with Win32 console programs such as interactive Python or node.js, but has a very limited default scroll-back, needs to be configured to use a Unicode font in order to display non-ASCII characters correctly, and prior to Windows 10 its window was not freely resizable and it only allowed rectangular text selections.

5.



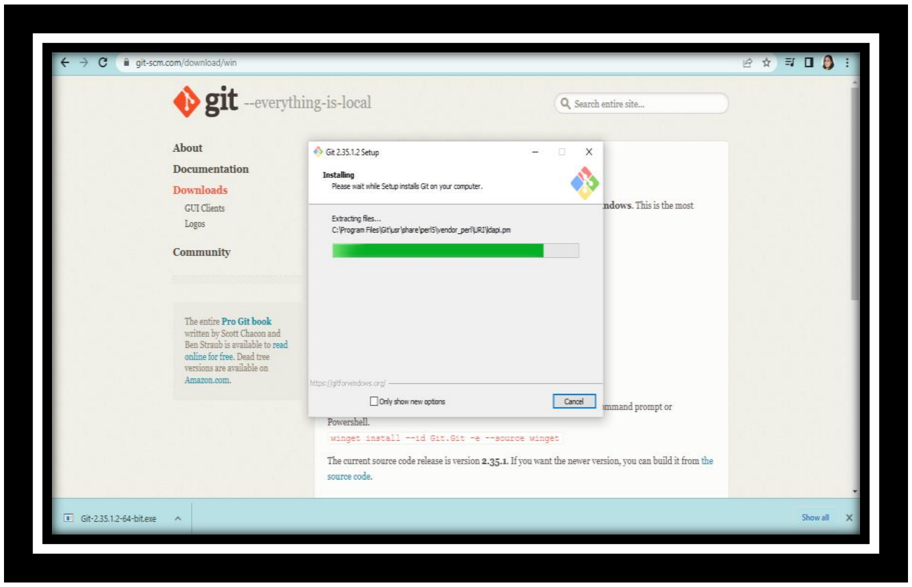
Git 2.35.1.2 Setup

Configuring experimental options
These features are developed actively. Would you like to try them?

☐ Enable experimental support for pseudo consoles.
(NEW!) This allows running native console programs like Node or Python in a Git Bash window without using winpty, but it still has known bugs.

☐ Enable experimental built-in file system monitor
(NEW!) Automatically run a built-in file system watcher, to speed up common operations such as `git status`, `git add`, `git commit`, etc in worktrees containing many files.

6.

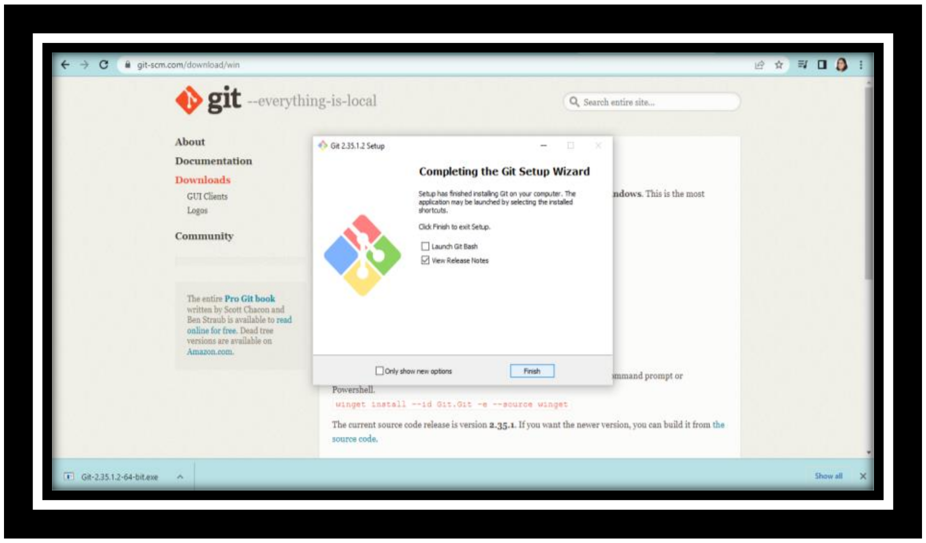


Git 2.35.1.2 Setup

Installing
Please wait while Setup installs Git on your computer.

Extracting files...
C:\Program Files\Git\usr\share\perl5\vendor_perl\URI\ldapi.pm

7.



Git 2.35.1.2 Setup

Completing the Git Setup Wizard

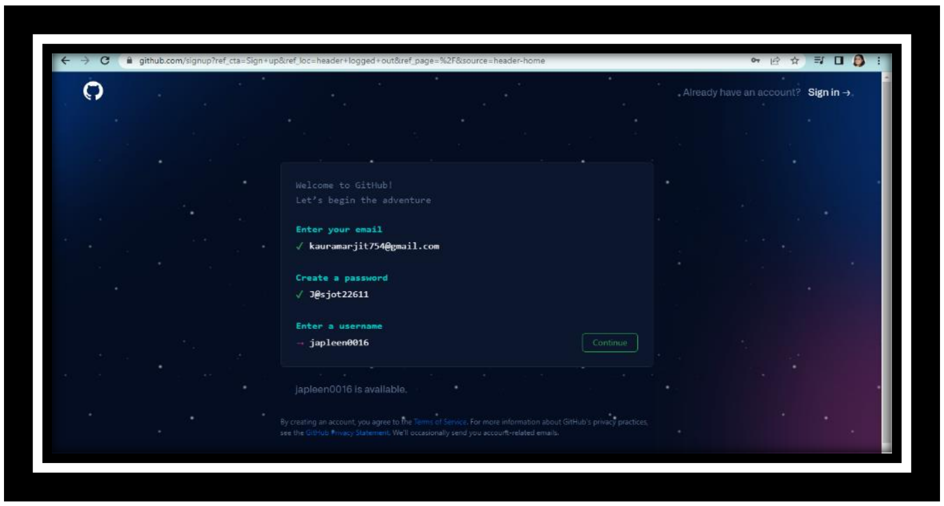
Setup has finished installing Git on your computer. The application may be launched by selecting the installed shortcuts.

Click Finish to exit Setup.

☐ Launch Git Bash
☑ View Release Notes

Step 3 – Setting up github account:

1.

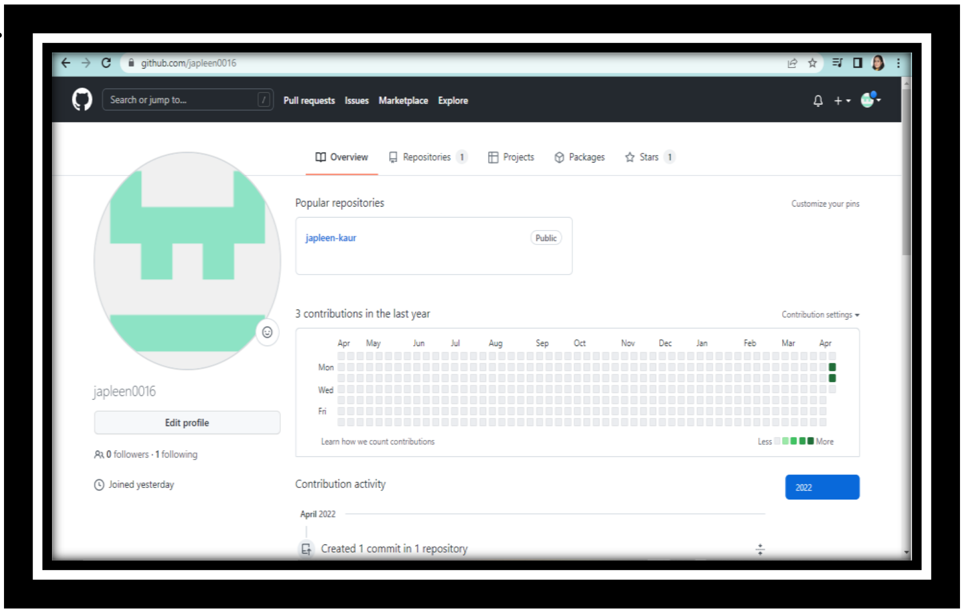


```
Welcome to GitHub!
Let's begin the adventure

Enter your email
✓ kauramarjit754@gmail.com

Create a password
✓ J@sjot22611

Enter a username
→ japleen0016
```

japleen0016 is available.

By creating an account, you agree to the Terms of Service. For more information about GitHub's privacy practices, see the GitHub Privacy Statement. We'll occasionally send you account-related emails.

2.



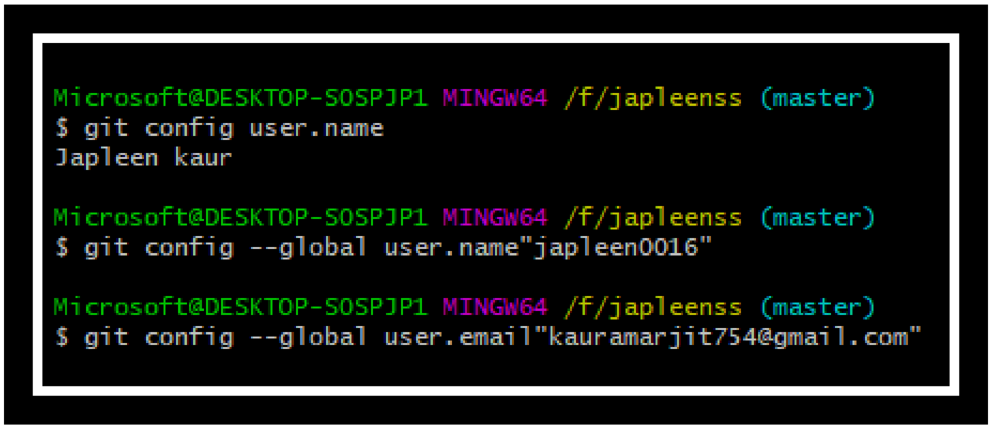
3. Configuration of Git:

Step1)

You can configure your Git by typing: -

1. Set your username: git config --global user.name "Your Name"
2. Set your email address: git config --global user.email "Your Email

The page looks like as: -



```
Microsoft@DESKTOP-SOSPJP1 MINGW64 /f/japleenss (master)
$ git config user.name
Japleen kaur

Microsoft@DESKTOP-SOSPJP1 MINGW64 /f/japleenss (master)
$ git config --global user.name"japleen0016"

Microsoft@DESKTOP-SOSPJP1 MINGW64 /f/japleenss (master)
$ git config --global user.email"kauramarjit754@gmail.com"
```

Step2)

You can check configuration of Git by typing -

1. **git config --list**

The page looks like as: -

4. Program to generate Logs:

Program to Generate logs:

Advantage of version control systems like git is that it can record changes. ‘Git log’ command is used to display all these changes that were made by the user in the repository. Log is a record of all the previous commits.

To understand Logs we need to get familiar with all the commands that are used in making changes to a repository.

1) **Repository: A repository is a directory that contains all the project-related data.**
2) **Git init: The git init command is used to create a new blank repository.**
3) **Git status: We can list all the untracked, modified and deleted files using the git status command.**
4) **Git add: Adds all the untracked and modified files to the staging area.**
5) **Git commit: Git commit finalizes the changes in the repository. Every commit is saved in the branch you are working in and can be used to revert back to older versions of the project.**

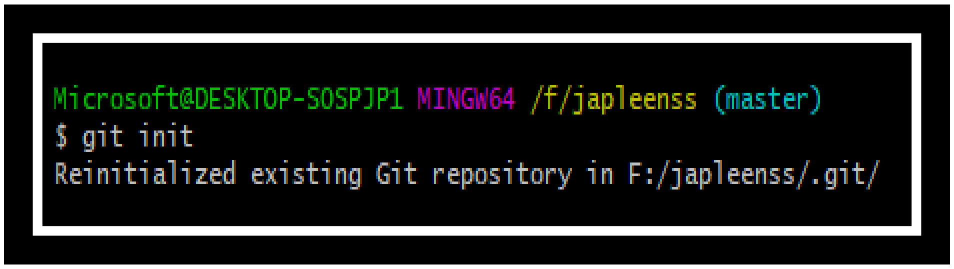
Making GIT Repository

Step1: GIT INIT

Initializing a new repository, You Can do it by typing: -

1. **git init**

The page looks like as: -

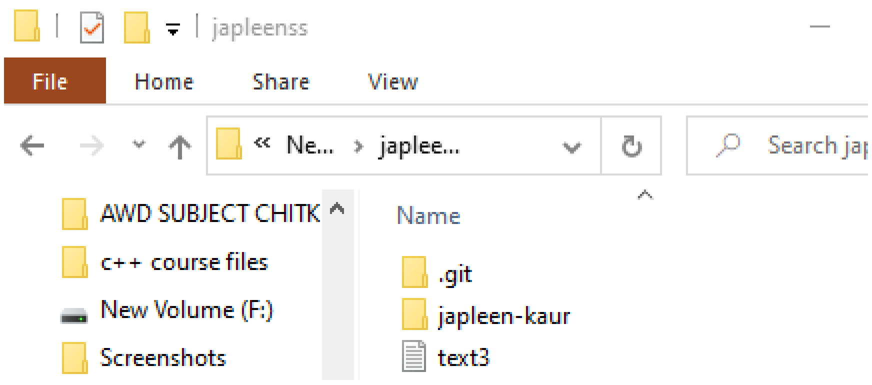


```
Microsoft@DESKTOP-SOSPJP1 MINGW64 /f/japleenss (master)
$ git init
Reinitialized existing Git repository in F:/japleenss/.git/
```

Step2: ADDING THE FILES TO THE FOLDER

Just like (Samples...)

The page looks like as: -



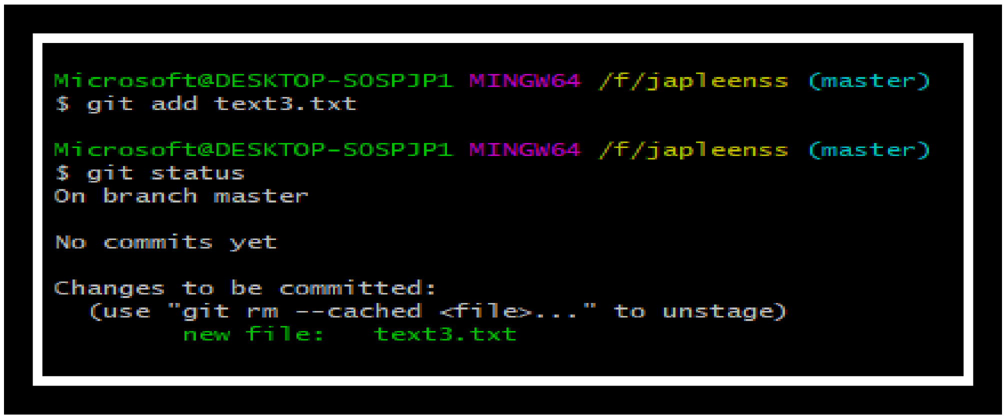
Step3: GIT ADD

The git add command adds a change in the working directory to the staging area.

You Can do it by typing: -

1. git add
2. git add (current file name)

The page looks like as: -



```
Microsoft@DESKTOP-SOSPJP1 MINGW64 /f/japleenss (master)
$ git add text3.txt

Microsoft@DESKTOP-SOSPJP1 MINGW64 /f/japleenss (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   text3.txt
```

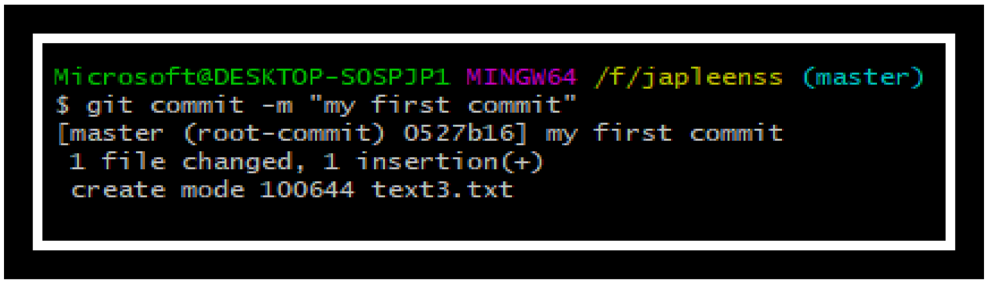
Step4: GIT COMMIT

The "commit" command is used to save your changes to the local repository.

You Can do it by typing: -

1. git commit -m"any text"

The page looks like as: -



```
Microsoft@DESKTOP-SOSPJP1 MINGW64 /f/japleenss (master)
$ git commit -m "my first commit"
[master (root-commit) 0527b16] my first commit
 1 file changed, 1 insertion(+)
 create mode 100644 text3.txt
```

Step5:GIT LOG

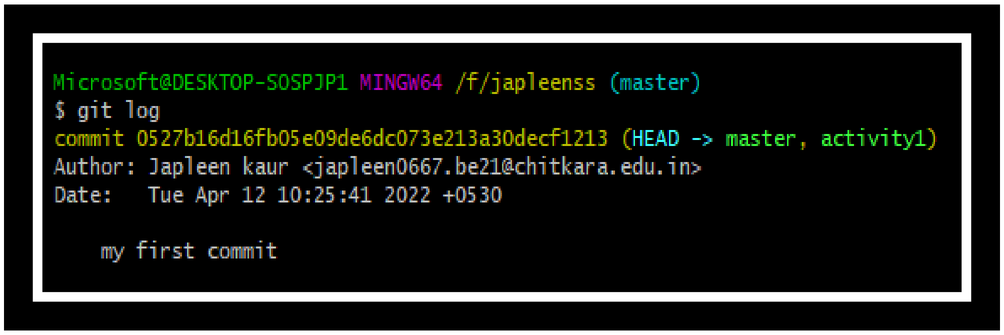
Git log will show all the commits made by the author with their time.
After every commit the checksum value (written In yellow color) of the folder changes.

Checksum is used to verify that the data in that file has not been tampered with or manipulated, possibly by a malicious entity.

You Can do it by typing: -

1. git log

The page looks like as: -



```
Microsoft@DESKTOP-SOSPJP1 MINGW64 /f/japleenss (master)
$ git log
commit 0527b16d16fb05e09de6dc073e213a30decf1213 (HEAD -> master, activity1)
Author: Japleen kaur <japleen0667.be21@chitkara.edu.in>
Date:   Tue Apr 12 10:25:41 2022 +0530

    my first commit
```

Step6:Create and Visualize Branches

A branch in Git is simply a lightweight movable pointer to one of these commits. The default branch name in Git is master.

5. Create and Visualize Branches

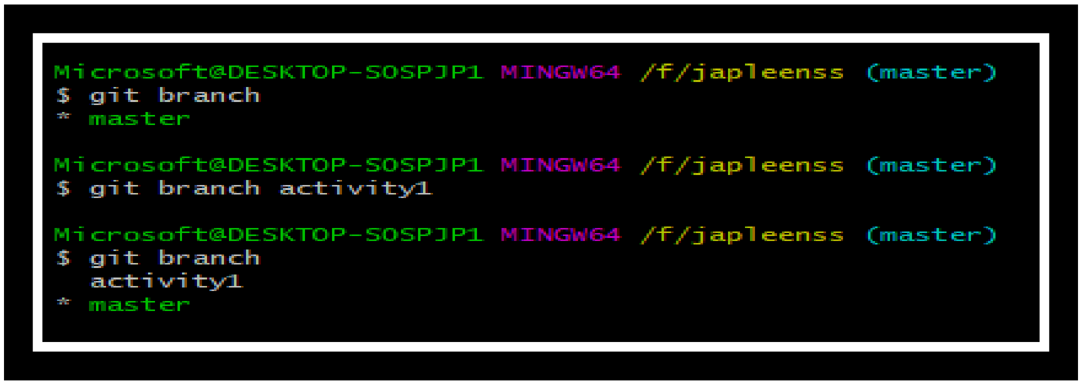
Step1:CHECKING UP THE BRANCHES

You can check which branch you are working in by using the command

1.’git branch’.

The default branch is always the master branch.

The page looks like as: -



```
Microsoft@DESKTOP-SOSPJP1 MINGW64 /f/japleenss (master)
$ git branch
* master

Microsoft@DESKTOP-SOSPJP1 MINGW64 /f/japleenss (master)
$ git branch activity1

Microsoft@DESKTOP-SOSPJP1 MINGW64 /f/japleenss (master)
$ git branch
  activity1
* master
```

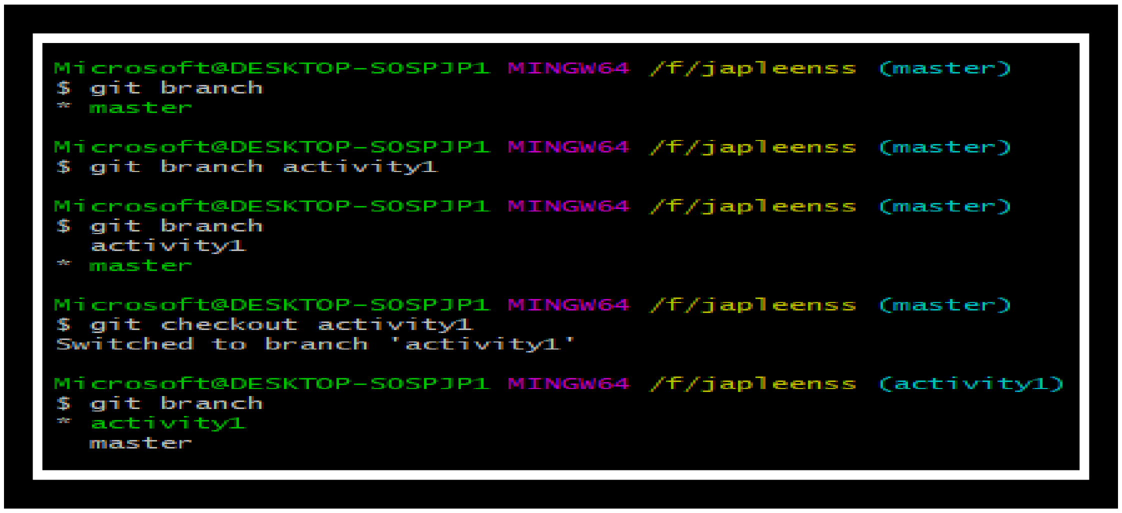
Step3:CHANGING BRANCHES

To switch to the other branch

You Can do it by typing: -

1. **git checkout (BRANCH NAME)**

The page looks like as: -



```
Microsoft@DESKTOP-SOSPJP1 MINGW64 /f/japleenss (master)
$ git branch
* master

Microsoft@DESKTOP-SOSPJP1 MINGW64 /f/japleenss (master)
$ git branch activity1

Microsoft@DESKTOP-SOSPJP1 MINGW64 /f/japleenss (master)
$ git branch
  activity1
* master

Microsoft@DESKTOP-SOSPJP1 MINGW64 /f/japleenss (master)
$ git checkout activity1
Switched to branch 'activity1'

Microsoft@DESKTOP-SOSPJP1 MINGW64 /f/japleenss (activity1)
$ git branch
* activity1
  master
```

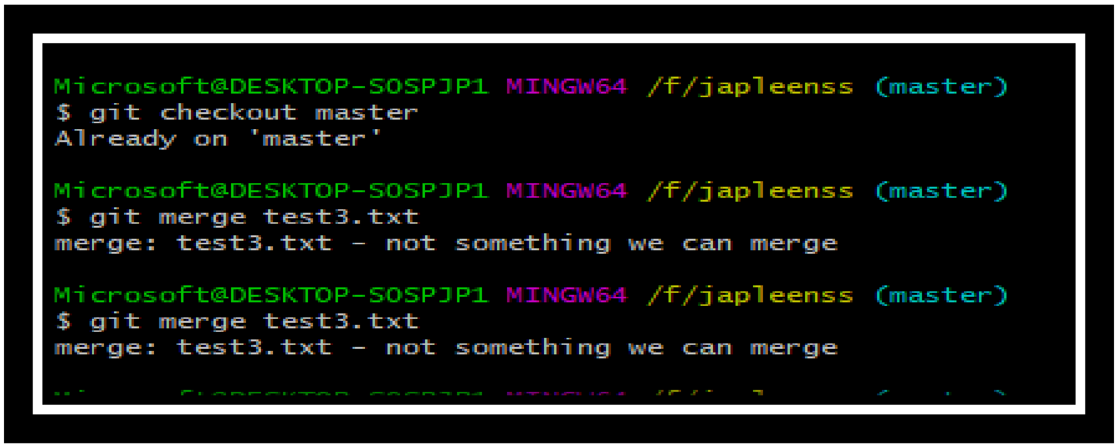
Step6:GIT MERGING

Now you can merge two branches by command.

1. **git merge (BRANCH NAME)**

If you want to merge a new branch in master branch you need to first checkout into the master branch and then run the command.

The page looks like as: -



```
Microsoft@DESKTOP-SOSPJP1 MINGW64 /f/japleenss (master)
$ git checkout master
Already on 'master'

Microsoft@DESKTOP-SOSPJP1 MINGW64 /f/japleenss (master)
$ git merge test3.txt
merge: test3.txt - not something we can merge

Microsoft@DESKTOP-SOSPJP1 MINGW64 /f/japleenss (master)
$ git merge test3.txt
merge: test3.txt - not something we can merge
```

6. Git Lifecycle Description:

There are three stages for git lifecycle:

1) **Working directory**
2) **Staging area**
3) **Git repository**

Working Directory:

The working directory is the folder in your local computer where the project files and folders are stored.

The local directory is created by the command 'git init' which creates a '.git' named folder which is used to track the files in the directory. '.git folder' is generally hidden but can be tracked enabling hidden files.

.git 3/28/2022 1:35 PM File folder

Staging area:

The staging area has those files which are supposed to go to the next commit. Only those files which are needed to go to the next commit stay

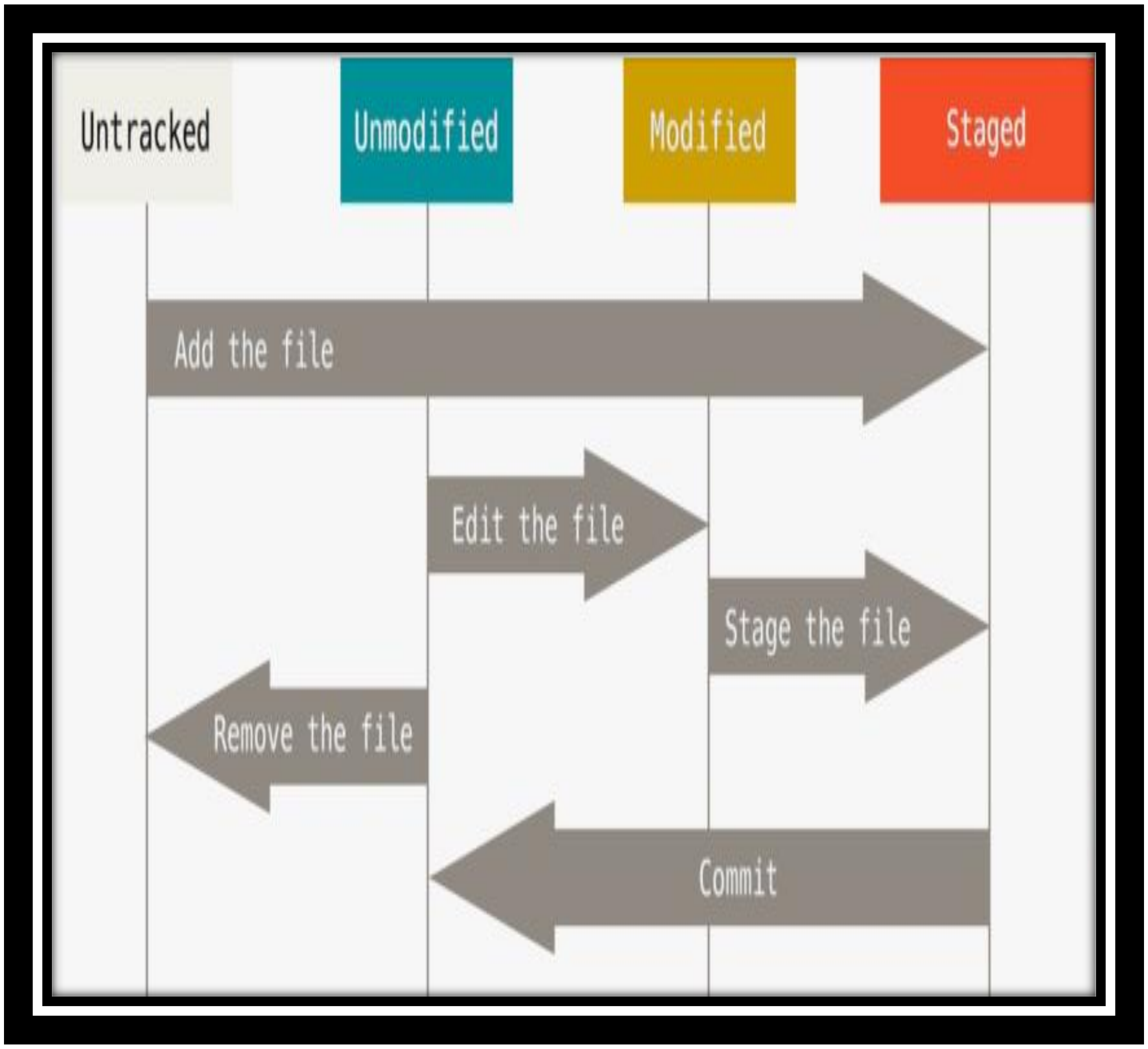
in the staging area.

You can shift the files to the git repository by using the command 'git add --a'.

<u>Git repository:</u>

Now since we have all the files that are to be tracked and are ready in the staging area, we are ready to commit our files using the git commit command. Commit helps us in keeping the track of the metadata of the files in our staging area. We specify every commit with a message which tells what the commit is about.

You can commit files by using command 'git commit -m "message"'

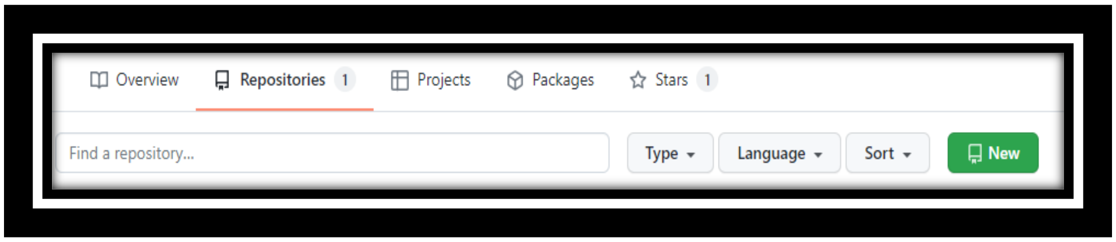


7. Uploading data on github

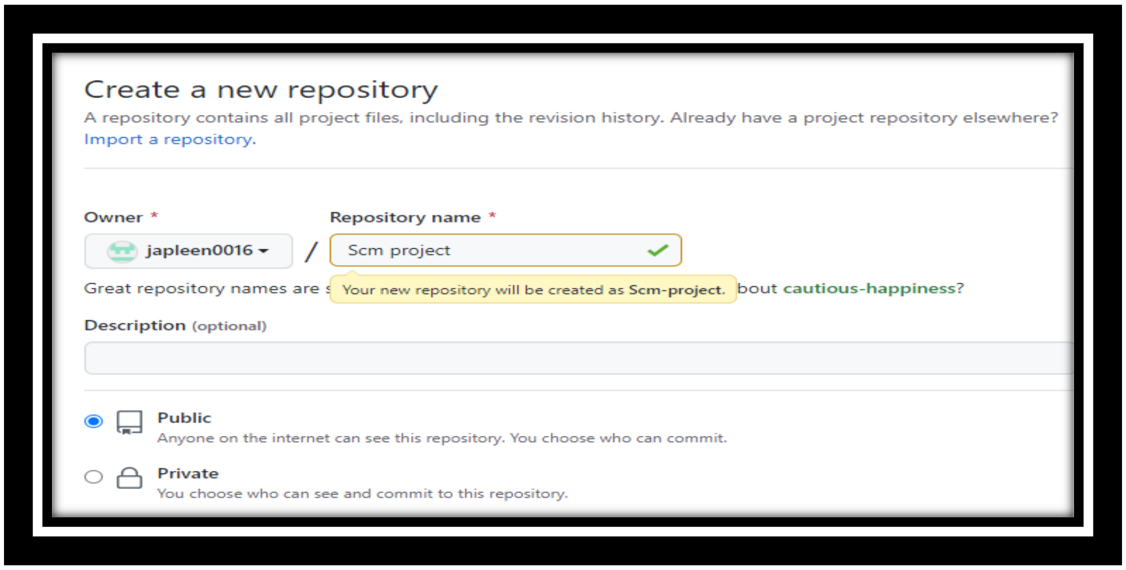
NOTE-
YOU HAVE TO MAKE A REPOSITORY IN GITHUB.

Step1) CREATING REPOSITORY IN GITHUB

The page looks like as: -



By clicking on new you are able to make a new repository.


Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? Import a repository.

Owner * japleen0016 / Repository name * Scm project

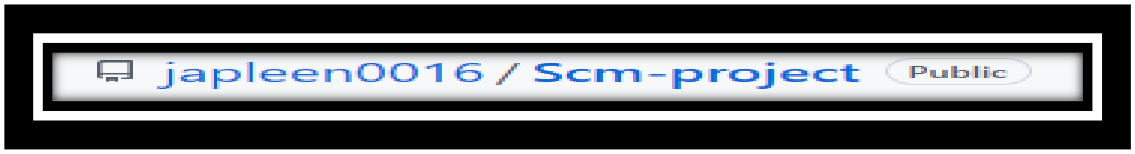
Great repository names are s Your new repository will be created as Scm-project. bout cautious-happiness?

Description (optional)

Public
Anyone on the internet can see this repository. You choose who can commit.

Private
You choose who can see and commit to this repository.

Write the repository name and click on next.



Your GITHUB Repository has been created.

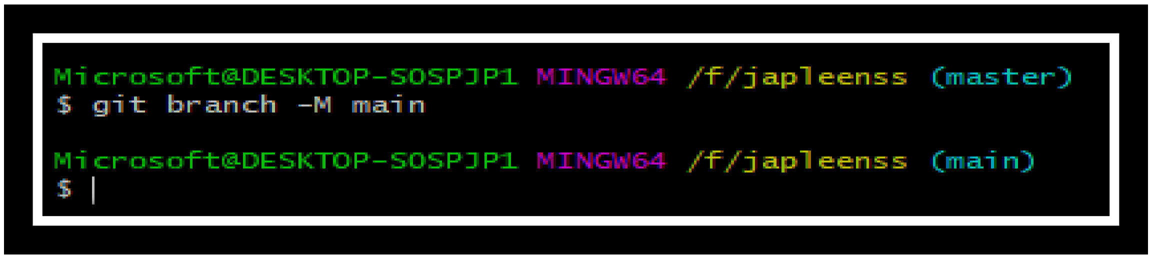
Step2) GIT ADDING REMOTE BRANCH

Git stores a branch as a reference to a commit, and a branch represents the tip of a series of commits.

You Can do it by typing: -

1. **git branch -M main**

The page looks like as: -



```
Microsoft@DESKTOP-SOSPJP1 MINGW64 /f/japleenss (master)
$ git branch -M main

Microsoft@DESKTOP-SOSPJP1 MINGW64 /f/japleenss (main)
$ |
```

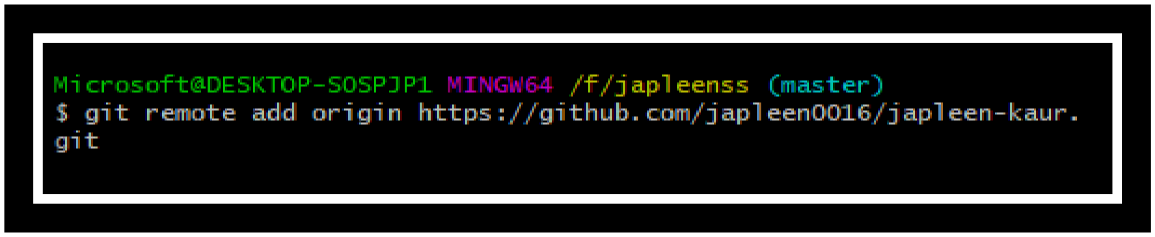
Step3) GIT ADDING REMOTE ORIGIN

Is a Git repository that's hosted on the Internet

You Can do it by typing: -

1. **git remote add origin (URL)**

The page looks like as: -



```
Microsoft@DESKTOP-SOSPJP1 MINGW64 /f/japleenss (master)
$ git remote add origin https://github.com/japleen0016/japleen-kaur.
git
```

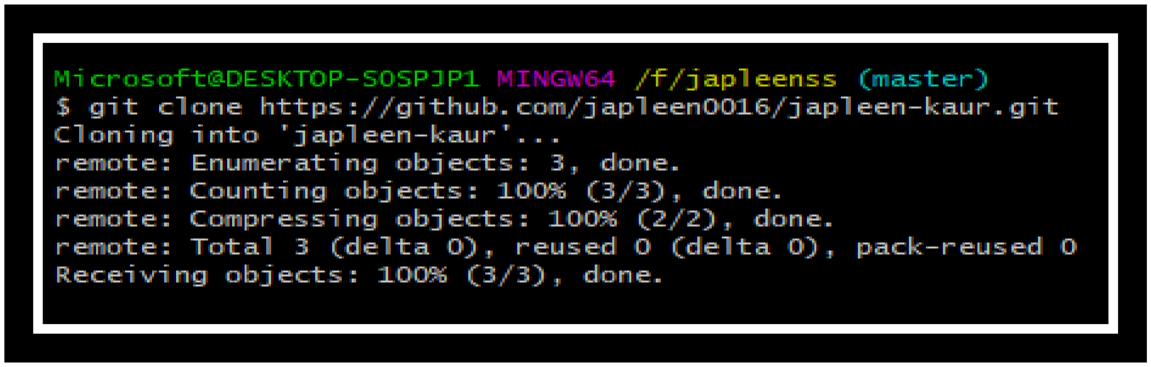
Step4) GIT CLONE

The git clone command is used to upload local repository content to a remote repository.

You Can do it by typing: -

1. **git clone url**

The page looks like as: -

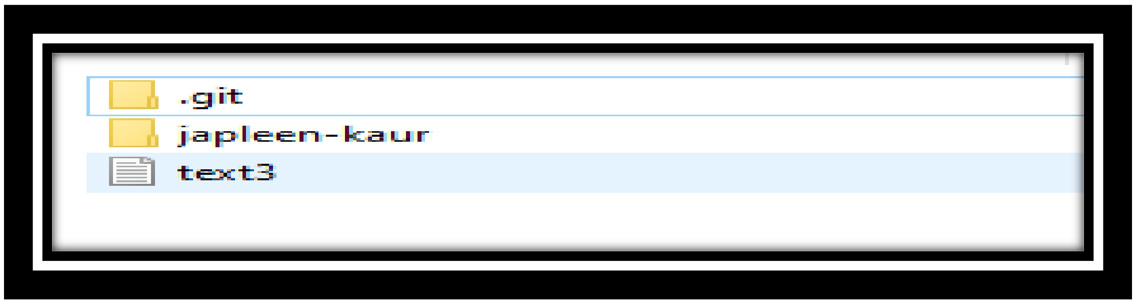


```
Microsoft@DESKTOP-SOSPJP1 MINGW64 /f/japleenss (master)
$ git clone https://github.com/japleen0016/japleen-kaur.git
Cloning into 'japleen-kaur'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (3/3), done.
```

Final Result

1. **Document in your system**

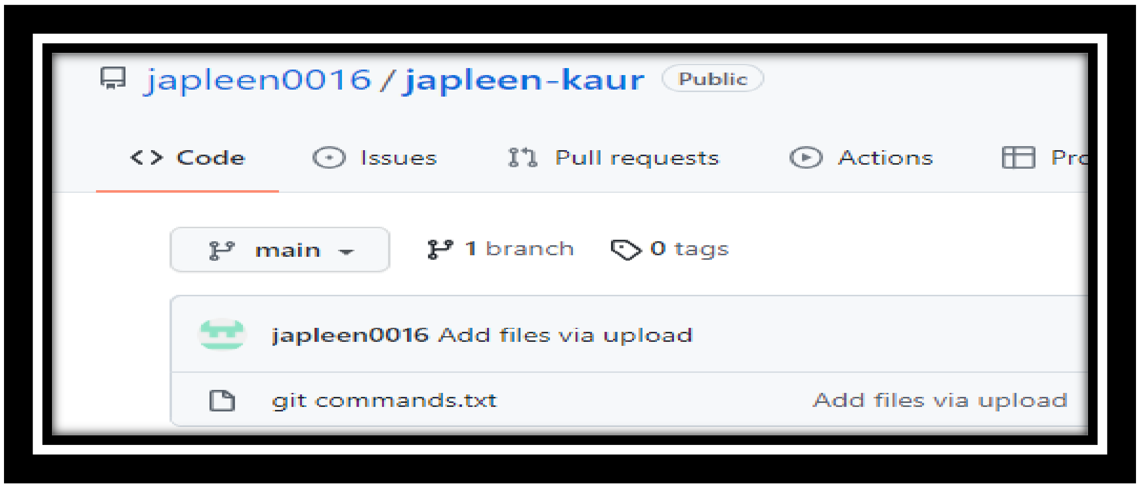
The page looks like as: -



```
.git
japleen-kaur
text3
```

2. **Document in your GITHUB Repository**

The page looks like as: -



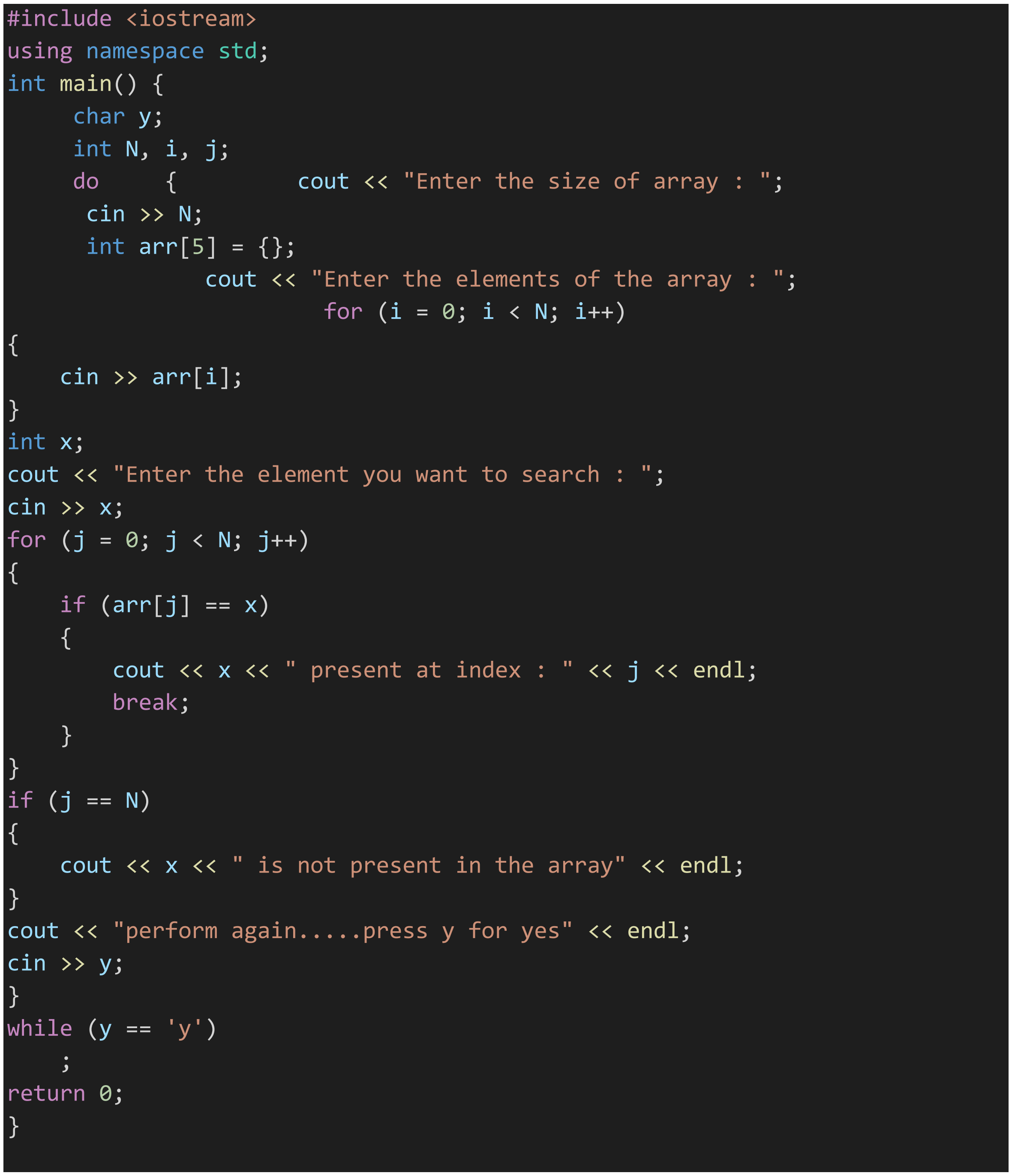
japleen0016 Add files via upload

git commands.txt Add files via upload

EXPERIMENT NO.2

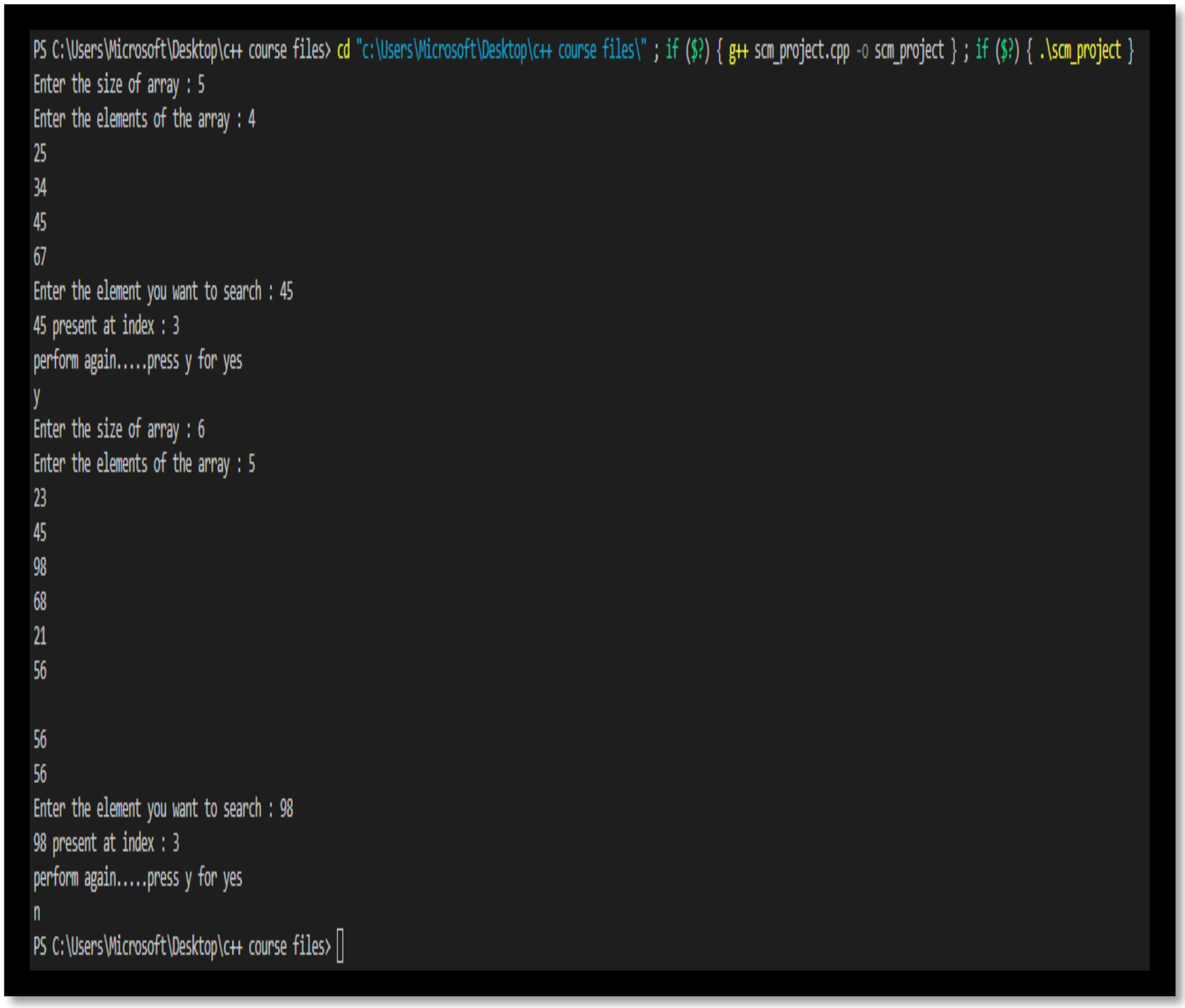
Aim: **Write a program to accept N numbers from the user and store them in an array. Then, accept another number from the user and search that using Linear Search.**

PROGRAM CODE:



```
#include <iostream>
using namespace std;
int main() {
     char y;
     int N, i, j;
     do     {          cout << "Enter the size of array : ";
      cin >> N;
      int arr[5] = {};
               cout << "Enter the elements of the array : ";
                        for (i = 0; i < N; i++)
{
    cin >> arr[i];
}
int x;
cout << "Enter the element you want to search : ";
cin >> x;
for (j = 0; j < N; j++)
{
    if (arr[j] == x)
    {
        cout << x << " present at index : " << j << endl;
        break;
    }
}
if (j == N)
{
    cout << x << " is not present in the array" << endl;
}
cout << "perform again.....press y for yes" << endl;
cin >> y;
}
while (y == 'y')
    ;
return 0;
}
```

OUTPUT:

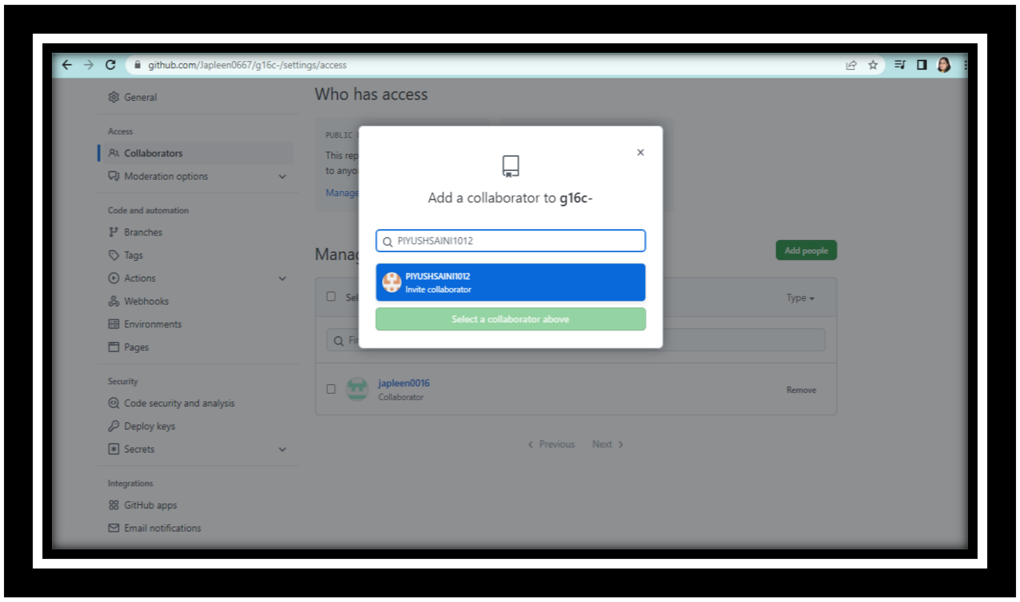


```
PS C:\Users\Microsoft\Desktop\c++ course files> cd "c:\Users\Microsoft\Desktop\c++ course files\" ; if ($?) { g++ scm_project.cpp -o scm_project } ; if ($?) { .\scm_project }
Enter the size of array : 5
Enter the elements of the array : 4
25
34
45
67
Enter the element you want to search : 45
45 present at index : 3
perform again.....press y for yes
y
Enter the size of array : 6
Enter the elements of the array : 5
23
45
98
68
21
56

56
56
Enter the element you want to search : 98
98 present at index : 3
perform again.....press y for yes
n
PS C:\Users\Microsoft\Desktop\c++ course files> 
```

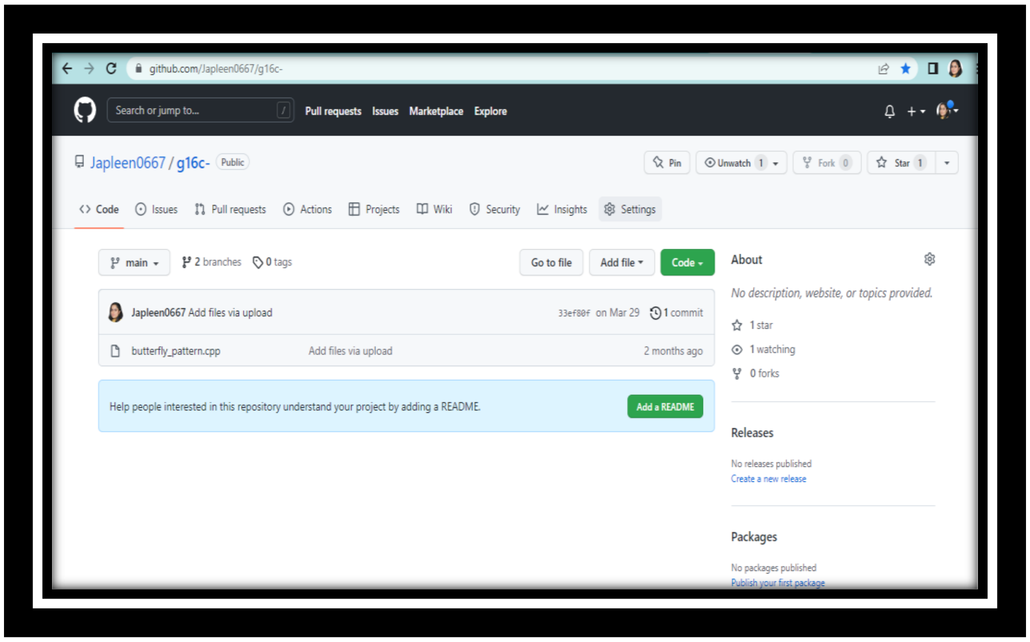
Task 1.2

1. Add Collaborators on Github Repo :-

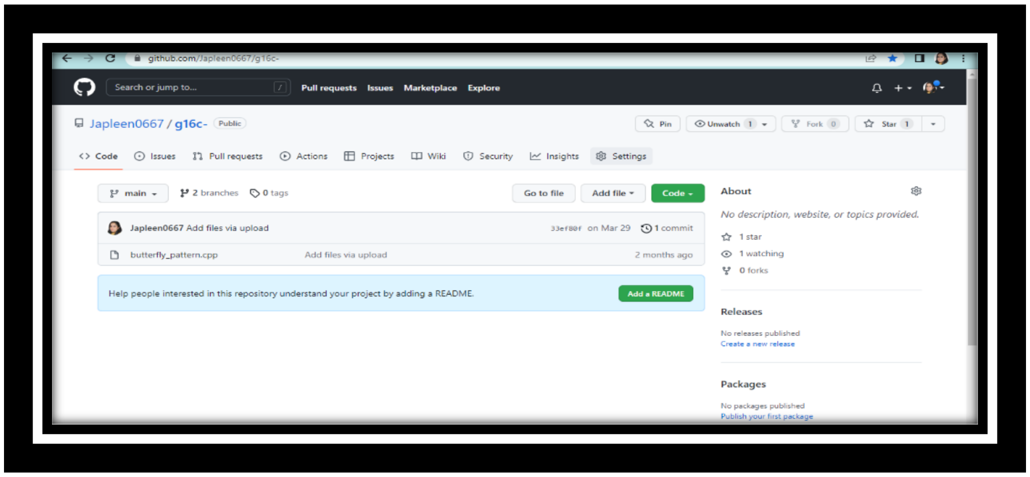
1. Ask for the username of the person you're inviting as a collaborator. ...



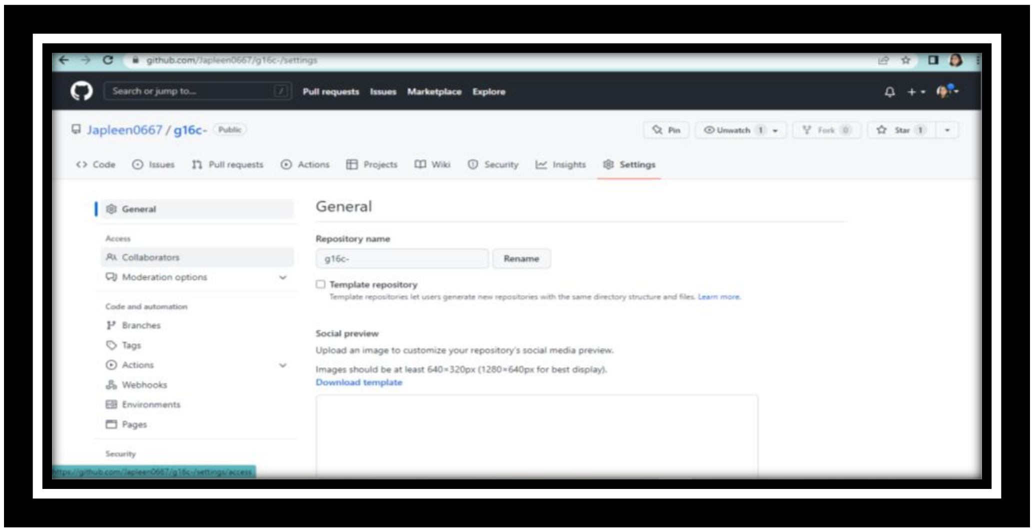
2. On GitHub.com, navigate to the main page of the repository.



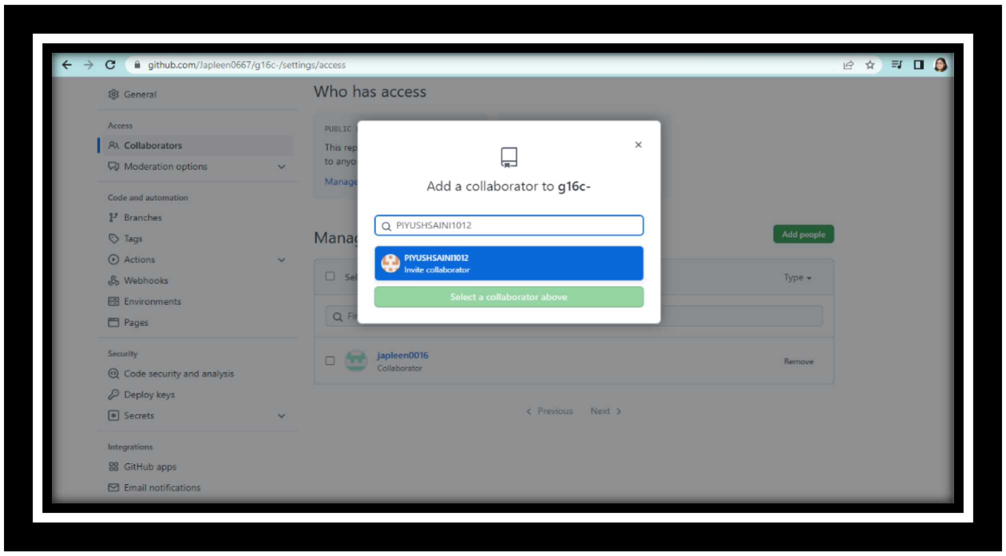
3. Under our repository name, click Settings

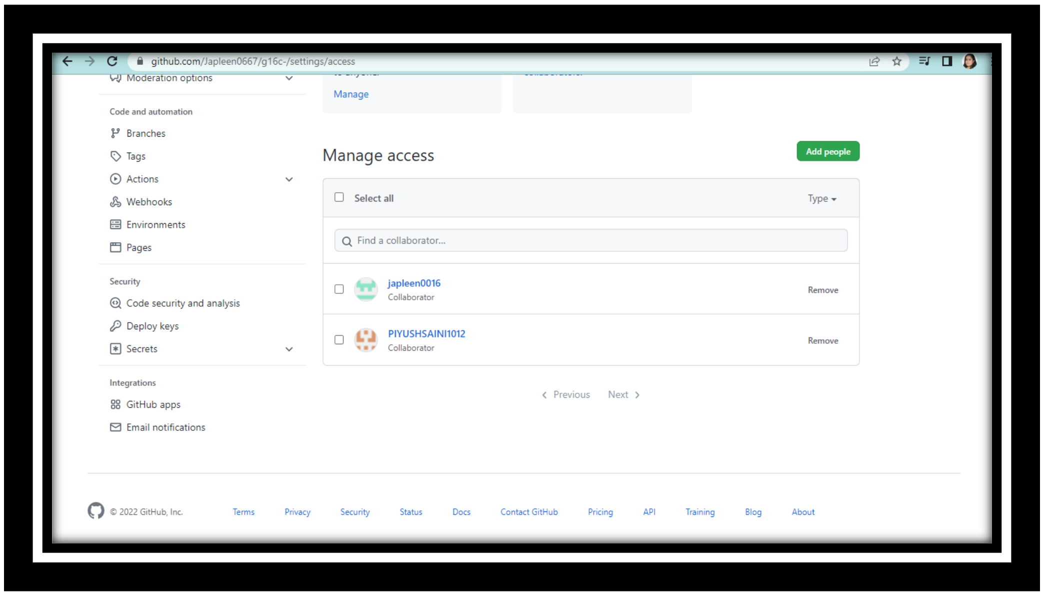


4. In the "Access" section of the sidebar, click Collaborators & teams.



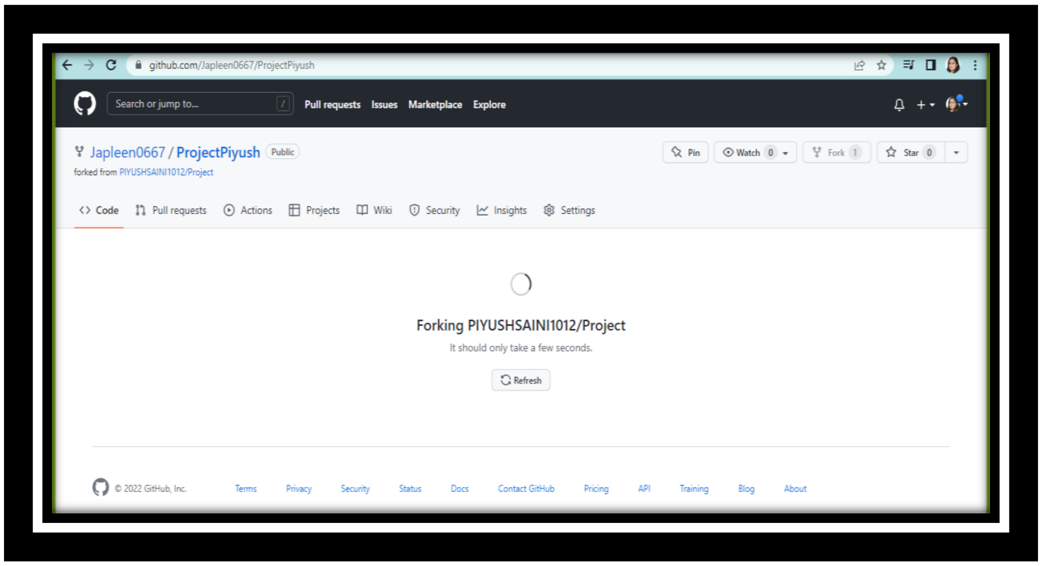
5. Click Invite a collaborator.

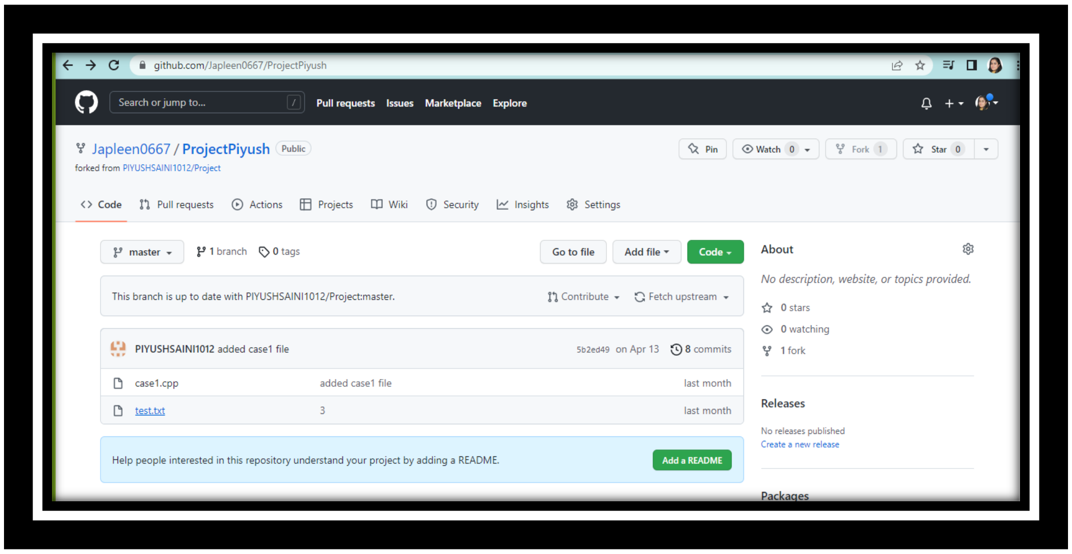




<u>2. Fork And Commit :-</u> A fork is a copy of a repository that we manage. Forks let us make changes to a project without affecting the original repository. We can fetch updates from or submit changes to the original repository with pull requests.

If we need to fork a GitHub or GitLab repo, it's as simple as navigating to the landing page of the repository in your web browser and clicking on the *Fork* button on the repository's home page. A forked copy of that Git repository will be added to your personal GitHub or GitLab repo. That's it. That's all we have to do to fork a Git repo.




github.com/Japleen0667/ProjectPiyush

Search or jump to... Pull requests Issues Marketplace Explore

Japleen0667 / ProjectPiyush Public
forked from PIYUSHSAINI1012/Project

Pin Watch 0 Fork 1 Star 0

Code Pull requests Actions Projects Wiki Security Insights Settings

master 1 branch 0 tags Go to file Add file Code

This branch is up to date with PIYUSHSAINI1012/Project:master. Contribute Fetch upstream

PIYUSHSAINI1012 added case1 file 5b2ed49 on Apr 13 8 commits

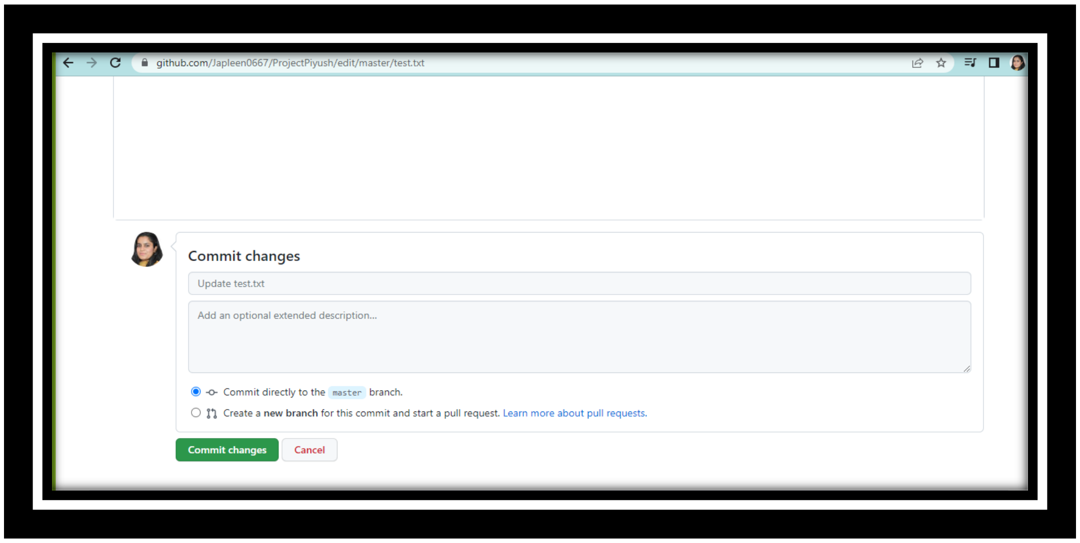
case1.cpp	added case1 file	last month
test.txt	3	last month

Help people interested in this repository understand your project by adding a README. Add a README

About
No description, website, or topics provided.
0 stars
0 watching
1 fork

Releases
No releases published
Create a new release

Packages


github.com/Japleen0667/ProjectPiyush/edit/master/test.txt

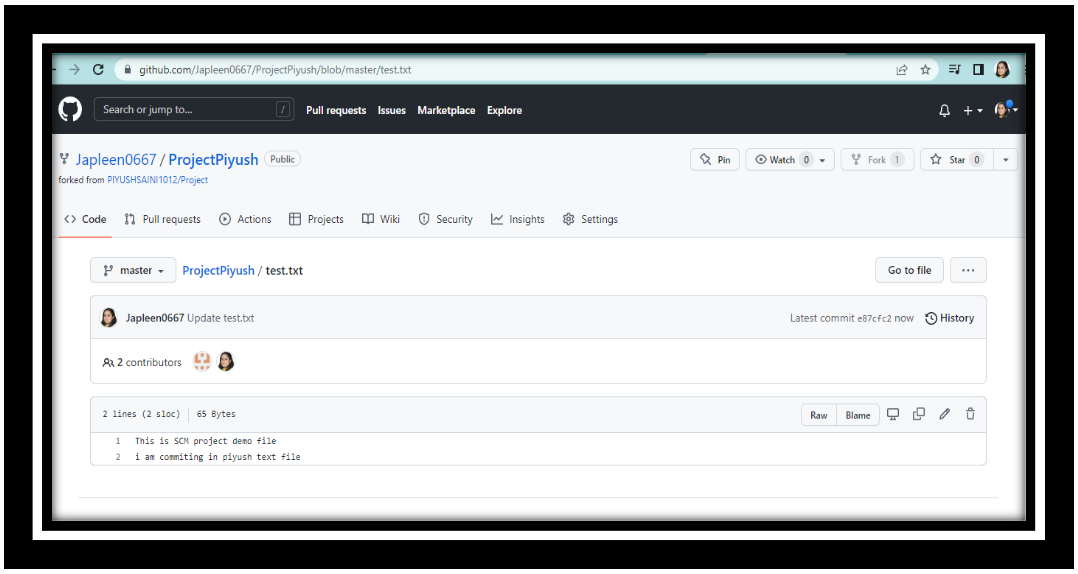
Commit changes

Update test.txt

Add an optional extended description...

Commit directly to the master branch.
Create a new branch for this commit and start a pull request. Learn more about pull requests.

Commit changes Cancel


github.com/Japleen0667/ProjectPiyush/blob/master/test.txt

Search or jump to... Pull requests Issues Marketplace Explore

Japleen0667 / ProjectPiyush Public
forked from PIYUSHSAINI1012/Project

Pin Watch 0 Fork 1 Star 0

Code Pull requests Actions Projects Wiki Security Insights Settings

master ProjectPiyush / test.txt Go to file

Japleen0667 Update test.txt Latest commit e87cfc2 now History

2 contributors

2 lines (2 sloc) | 65 Bytes Raw Blame

```
This is SCM project demo file
i am commiting in piyush text file
```

3. Merge and Resolve conflicts created due to own Activity and Collaborators activity:-

There are a few steps that could reduce the steps needed to resolve merge conflicts in Git.

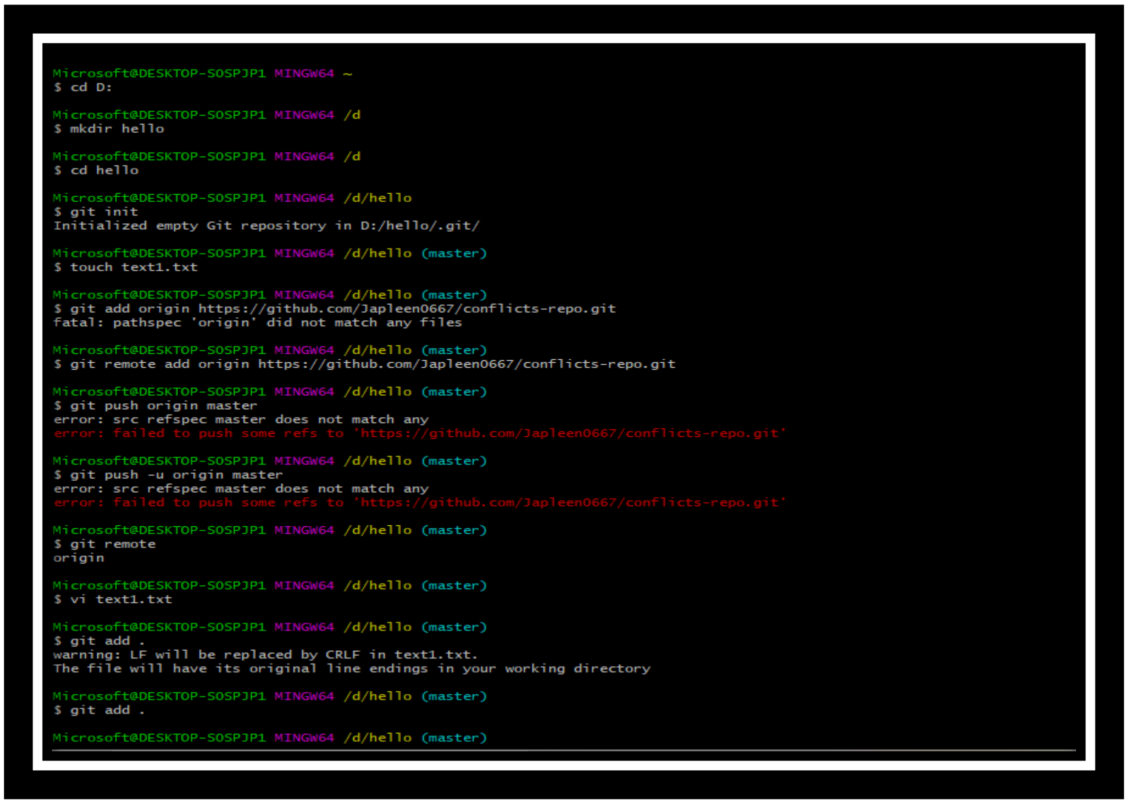
1. The easiest way to resolve a conflicted file is to open it and make any necessary changes.

2. After editing the file, we can use the git add a command to stage the new merged content.

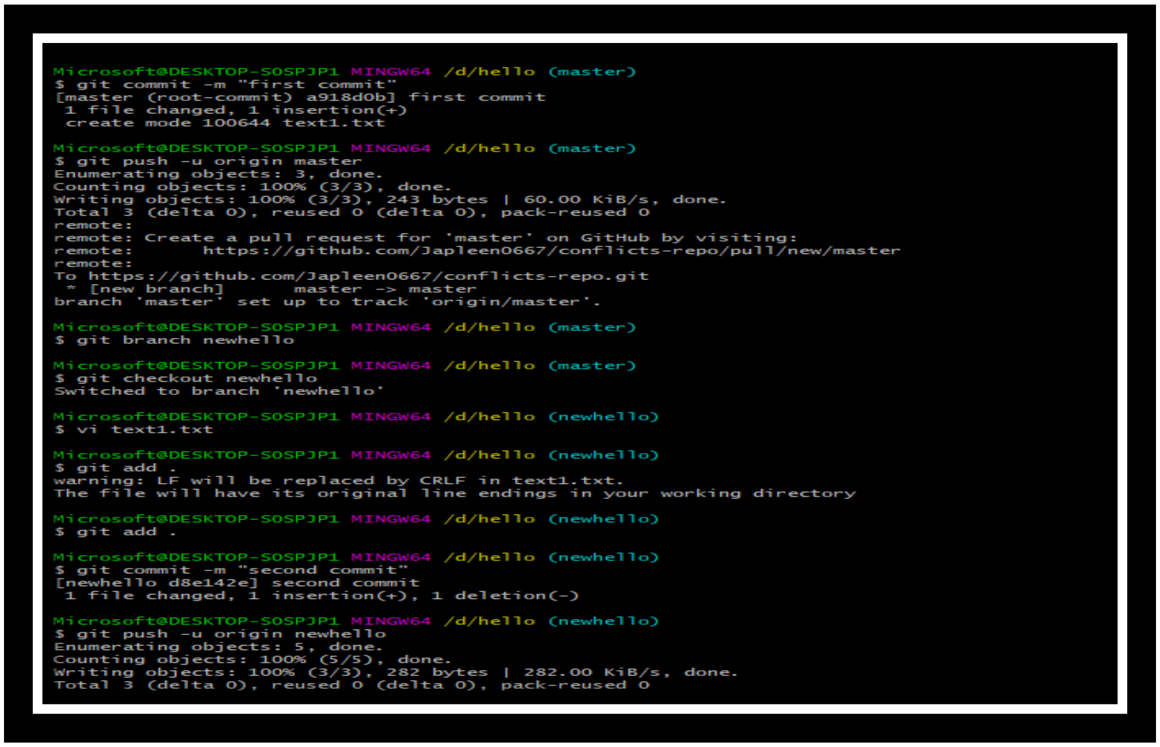
3. The final step is to create a new commit with the help of the git commit command.

4. Git will create a new merge commit to finalize the merge

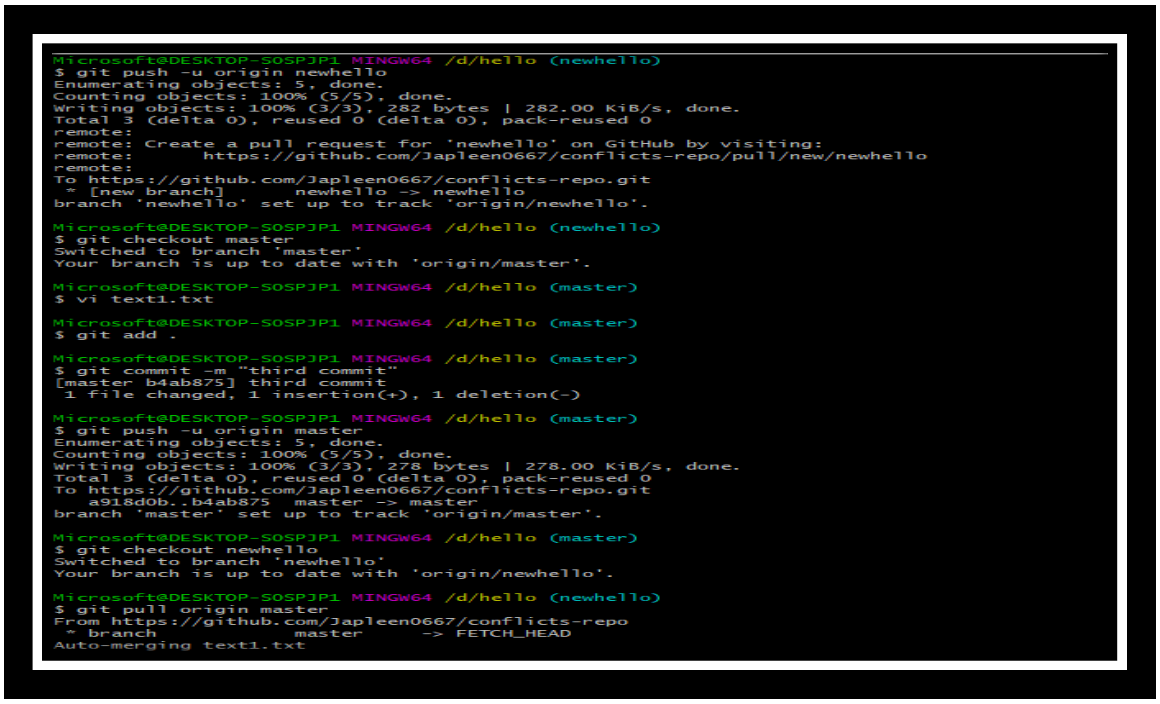
Let us now look into the Git commands that may play a significant role in resolving conflicts.



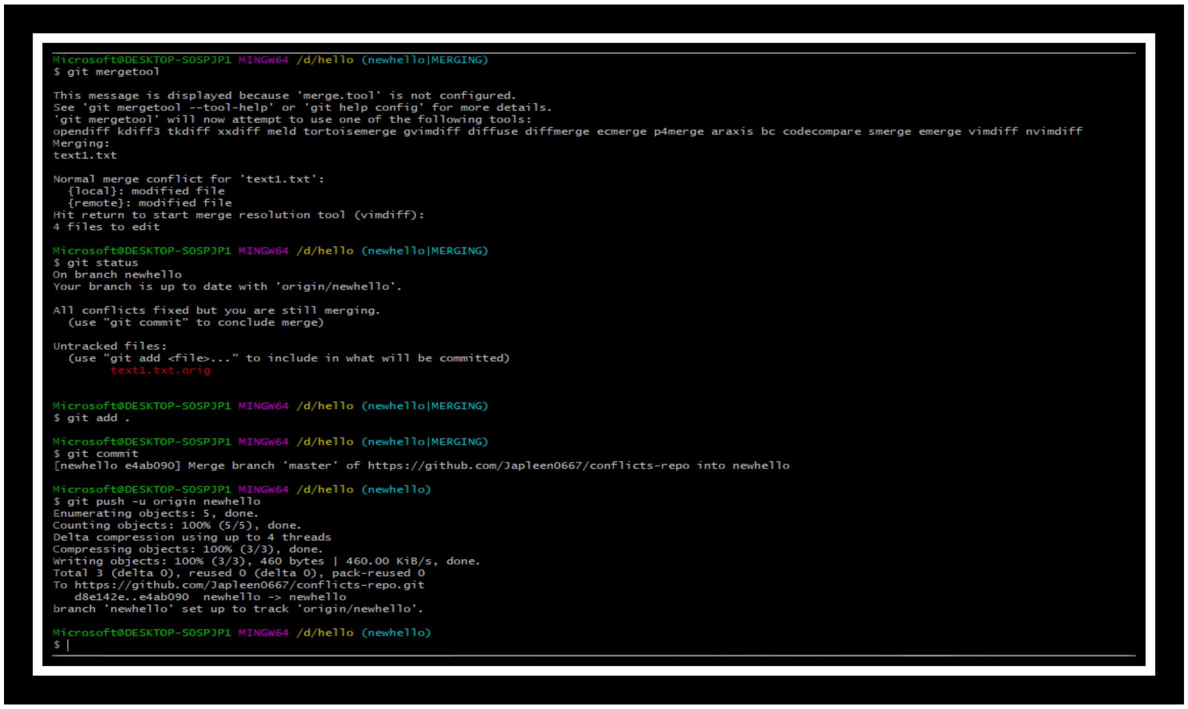
```
Microsoft@DESKTOP-SOSPJP1 MINGW64 ~
$ cd D:

Microsoft@DESKTOP-SOSPJP1 MINGW64 /d
$ mkdir hello

Microsoft@DESKTOP-SOSPJP1 MINGW64 /d
$ cd hello

Microsoft@DESKTOP-SOSPJP1 MINGW64 /d/hello
$ git init
Initialized empty Git repository in D:/hello/.git/

Microsoft@DESKTOP-SOSPJP1 MINGW64 /d/hello (master)
$ touch text1.txt

Microsoft@DESKTOP-SOSPJP1 MINGW64 /d/hello (master)
$ git add origin https://github.com/Japleen0667/conflicts-repo.git
fatal: pathspec 'origin' did not match any files

Microsoft@DESKTOP-SOSPJP1 MINGW64 /d/hello (master)
$ git remote add origin https://github.com/Japleen0667/conflicts-repo.git

Microsoft@DESKTOP-SOSPJP1 MINGW64 /d/hello (master)
$ git push origin master
error: src refspec master does not match any
error: failed to push some refs to 'https://github.com/Japleen0667/conflicts-repo.git'

Microsoft@DESKTOP-SOSPJP1 MINGW64 /d/hello (master)
$ git push -u origin master
error: src refspec master does not match any
error: failed to push some refs to 'https://github.com/Japleen0667/conflicts-repo.git'

Microsoft@DESKTOP-SOSPJP1 MINGW64 /d/hello (master)
$ git remote
origin

Microsoft@DESKTOP-SOSPJP1 MINGW64 /d/hello (master)
$ vi text1.txt

Microsoft@DESKTOP-SOSPJP1 MINGW64 /d/hello (master)
$ git add .
warning: LF will be replaced by CRLF in text1.txt.
The file will have its original line endings in your working directory

Microsoft@DESKTOP-SOSPJP1 MINGW64 /d/hello (master)
$ git add .

Microsoft@DESKTOP-SOSPJP1 MINGW64 /d/hello (master)
```



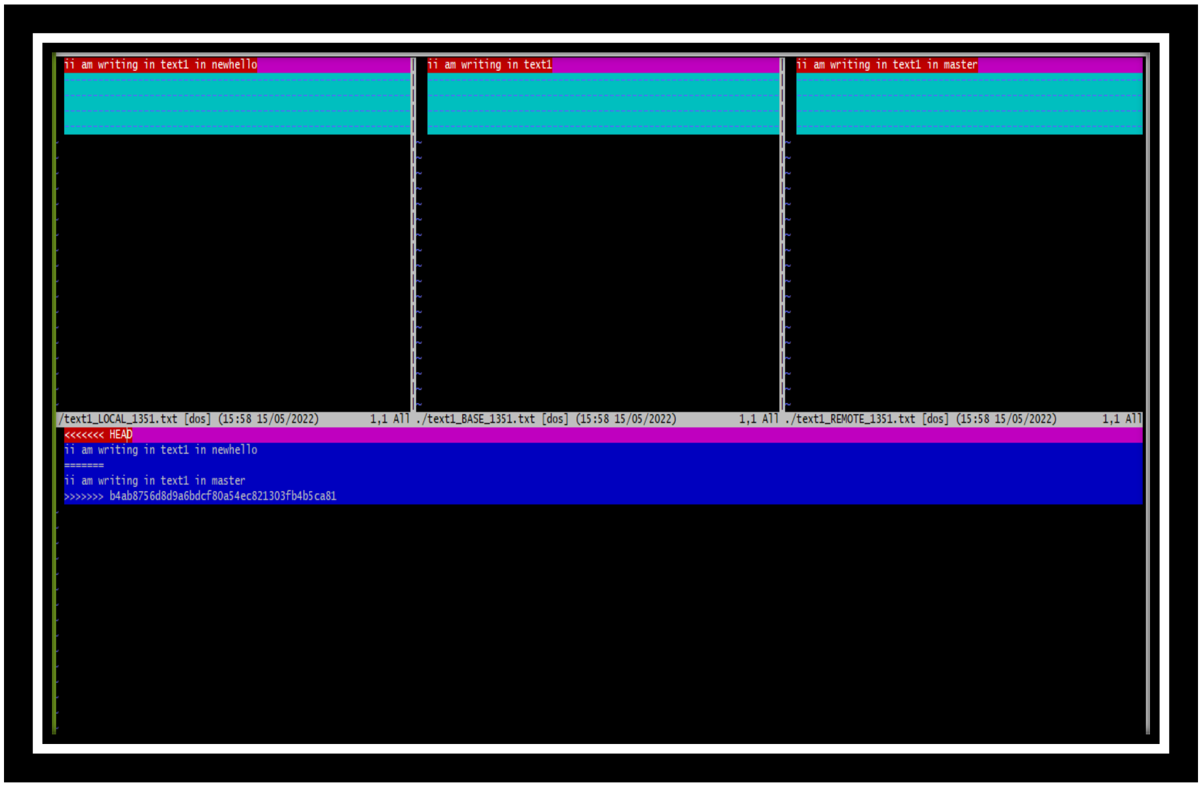
```
Microsoft@DESKTOP-SOSPJP1 MINGW64 /d/hello (master)
$ git commit -m "first commit"
[master (root-commit) a918d0b] first commit
 1 file changed, 1 insertion(+)
 create mode 100644 text1.txt

Microsoft@DESKTOP-SOSPJP1 MINGW64 /d/hello (master)
$ git push -u origin master
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 243 bytes | 60.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'master' on GitHub by visiting:
remote:      https://github.com/Japleen0667/conflicts-repo/pull/new/master
remote:
To https://github.com/Japleen0667/conflicts-repo.git
 * [new branch]      master -> master
branch 'master' set up to track 'origin/master'.

Microsoft@DESKTOP-SOSPJP1 MINGW64 /d/hello (master)
$ git branch newhello

Microsoft@DESKTOP-SOSPJP1 MINGW64 /d/hello (master)
$ git checkout newhello
Switched to branch 'newhello'

Microsoft@DESKTOP-SOSPJP1 MINGW64 /d/hello (newhello)
$ vi text1.txt

Microsoft@DESKTOP-SOSPJP1 MINGW64 /d/hello (newhello)
$ git add .
warning: LF will be replaced by CRLF in text1.txt.
The file will have its original line endings in your working directory

Microsoft@DESKTOP-SOSPJP1 MINGW64 /d/hello (newhello)
$ git add .

Microsoft@DESKTOP-SOSPJP1 MINGW64 /d/hello (newhello)
$ git commit -m "second commit"
[newhello d8e142e] second commit
 1 file changed, 1 insertion(+), 1 deletion(-)

Microsoft@DESKTOP-SOSPJP1 MINGW64 /d/hello (newhello)
$ git push -u origin newhello
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Writing objects: 100% (3/3), 282 bytes | 282.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
```



```
Microsoft@DESKTOP-SOSPJP1 MINGW64 /d/hello (newhello)
$ git push -u origin newhello
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Writing objects: 100% (3/3), 282 bytes | 282.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'newhello' on GitHub by visiting:
remote:      https://github.com/Japleen0667/conflicts-repo/pull/new/newhello
remote:
To https://github.com/Japleen0667/conflicts-repo.git
 * [new branch]      newhello -> newhello
branch 'newhello' set up to track 'origin/newhello'.

Microsoft@DESKTOP-SOSPJP1 MINGW64 /d/hello (newhello)
$ git checkout master
Switched to branch 'master'
Your branch is up to date with 'origin/master'.

Microsoft@DESKTOP-SOSPJP1 MINGW64 /d/hello (master)
$ vi text1.txt

Microsoft@DESKTOP-SOSPJP1 MINGW64 /d/hello (master)
$ git add .

Microsoft@DESKTOP-SOSPJP1 MINGW64 /d/hello (master)
$ git commit -m "third commit"
[master b4ab875] third commit
 1 file changed, 1 insertion(+), 1 deletion(-)

Microsoft@DESKTOP-SOSPJP1 MINGW64 /d/hello (master)
$ git push -u origin master
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Writing objects: 100% (3/3), 278 bytes | 278.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/Japleen0667/conflicts-repo.git
   a918d0b..b4ab875  master -> master
branch 'master' set up to track 'origin/master'.

Microsoft@DESKTOP-SOSPJP1 MINGW64 /d/hello (master)
$ git checkout newhello
Switched to branch 'newhello'
Your branch is up to date with 'origin/newhello'.

Microsoft@DESKTOP-SOSPJP1 MINGW64 /d/hello (newhello)
$ git pull origin master
From https://github.com/Japleen0667/conflicts-repo
 * branch            master     -> FETCH_HEAD
Auto-merging text1.txt
```



```
Microsoft@DESKTOP-SOSPJP1 MINGW64 /d/hello (newhello|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:
text1.txt

Normal merge conflict for 'text1.txt':
  {local}: modified file
  {remote}: modified file
Hit return to start merge resolution tool (vimdiff):
4 files to edit

Microsoft@DESKTOP-SOSPJP1 MINGW64 /d/hello (newhello|MERGING)
$ git status
On branch newhello
Your branch is up to date with 'origin/newhello'.

All conflicts fixed but you are still merging.
  (use "git commit" to conclude merge)

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        text1.txt.orig


Microsoft@DESKTOP-SOSPJP1 MINGW64 /d/hello (newhello|MERGING)
$ git add .

Microsoft@DESKTOP-SOSPJP1 MINGW64 /d/hello (newhello|MERGING)
$ git commit
[newhello e4ab090] Merge branch 'master' of https://github.com/Japleen0667/conflicts-repo into newhello

Microsoft@DESKTOP-SOSPJP1 MINGW64 /d/hello (newhello)
$ git push -u origin newhello
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), 460 bytes | 460.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/Japleen0667/conflicts-repo.git
   d8e142e..e4ab090  newhello -> newhello
branch 'newhello' set up to track 'origin/newhello'.

Microsoft@DESKTOP-SOSPJP1 MINGW64 /d/hello (newhello)
$ |
```

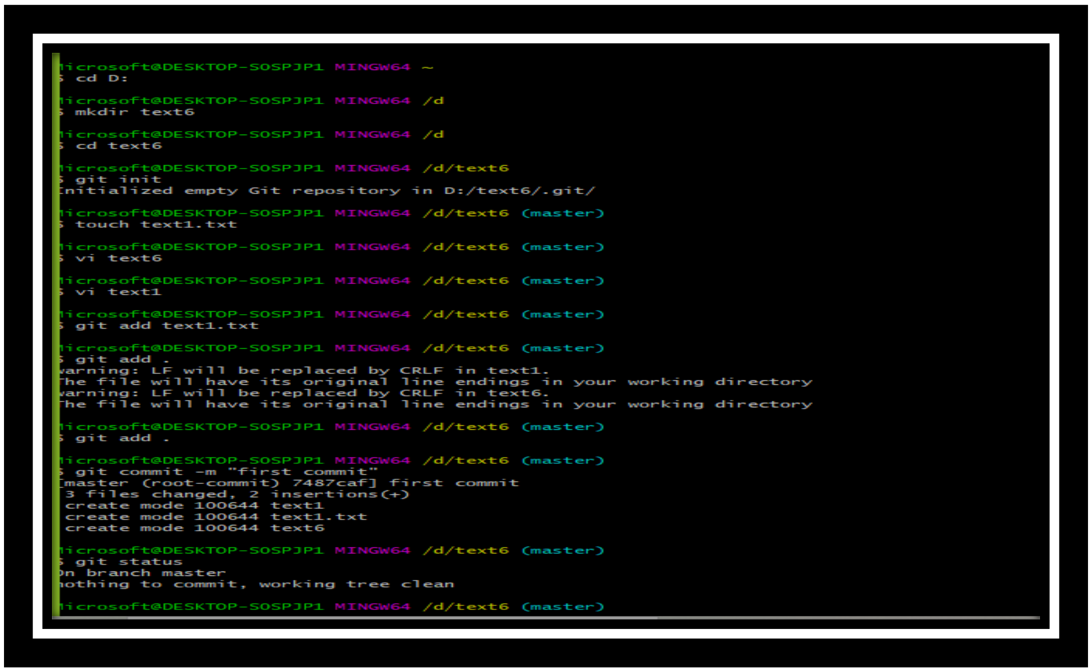


```
ii am writing in text1 in newhello   | ii am writing in text1   | ii am writing in text1 in master
/text1_LOCAL_1351.txt [dos] (15:58 15/05/2022)   1,1 All ./text1_BASE_1351.txt [dos] (15:58 15/05/2022)   1,1 All ./text1_REMOTE_1351.txt [dos] (15:58 15/05/2022)   1,1 All
<<<<<<< HEAD
ii am writing in text1 in newhello
=======
ii am writing in text1 in master
>>>>>>> b4ab8756d8d9a6bdcf80a54ec821303fb4b5ca81
```

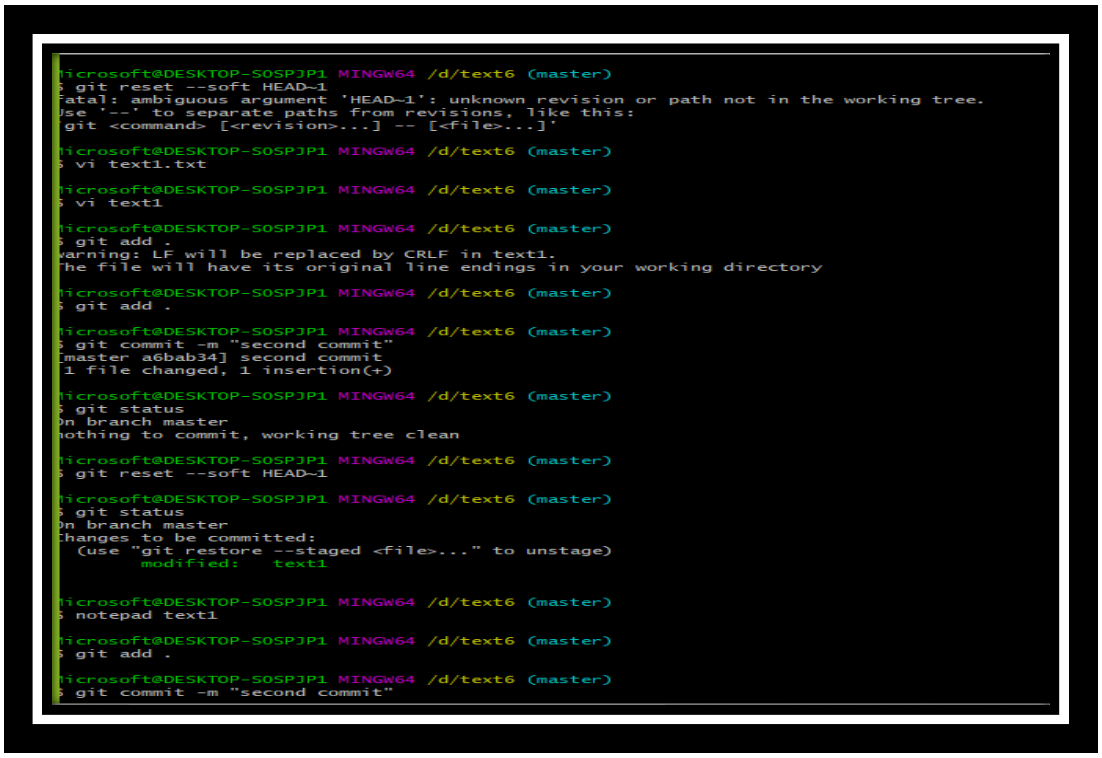
4. <u>Git Reset :-</u> Git reset is a powerful command that is used to undo local changes to the state of a Git repo. Git reset operates on "The Three Trees of Git". These trees are the Commit History (HEAD), the Staging Index, and the Working Directory.

The easiest way to undo the last Git commit is to execute the “git reset” command with the “–soft” option that will preserve changes done to your files.

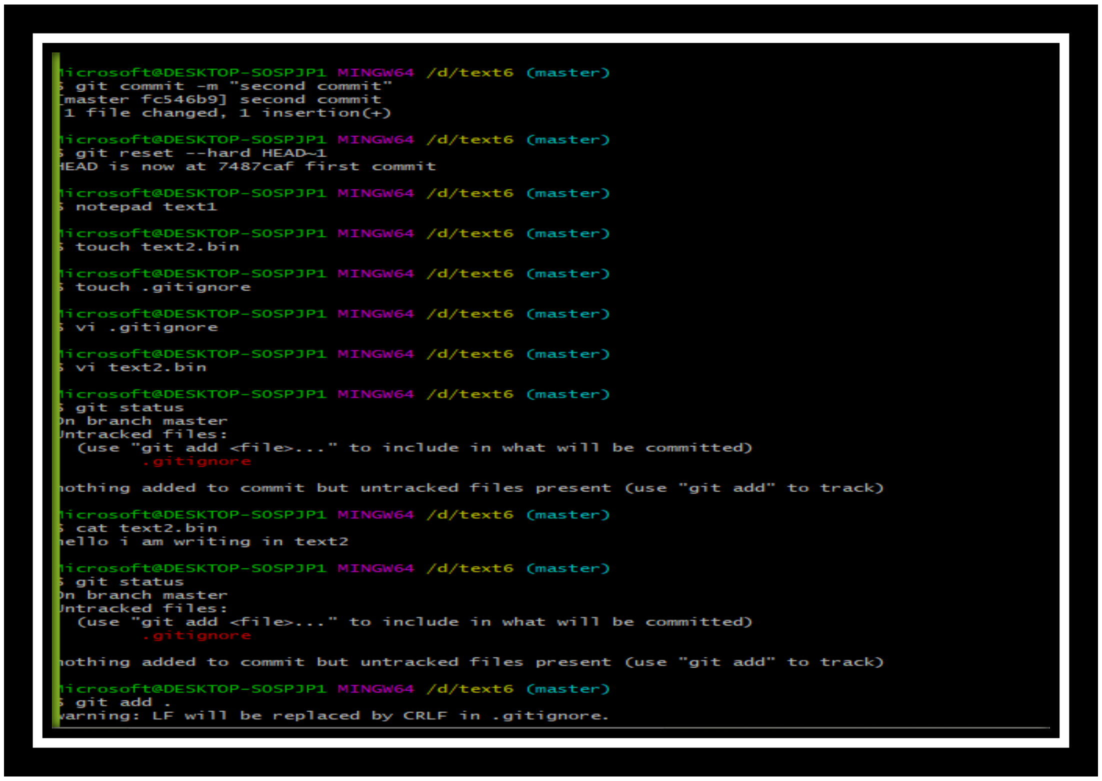
Git reset --hard , which will completely destroy any changes and remove them from the local directory.



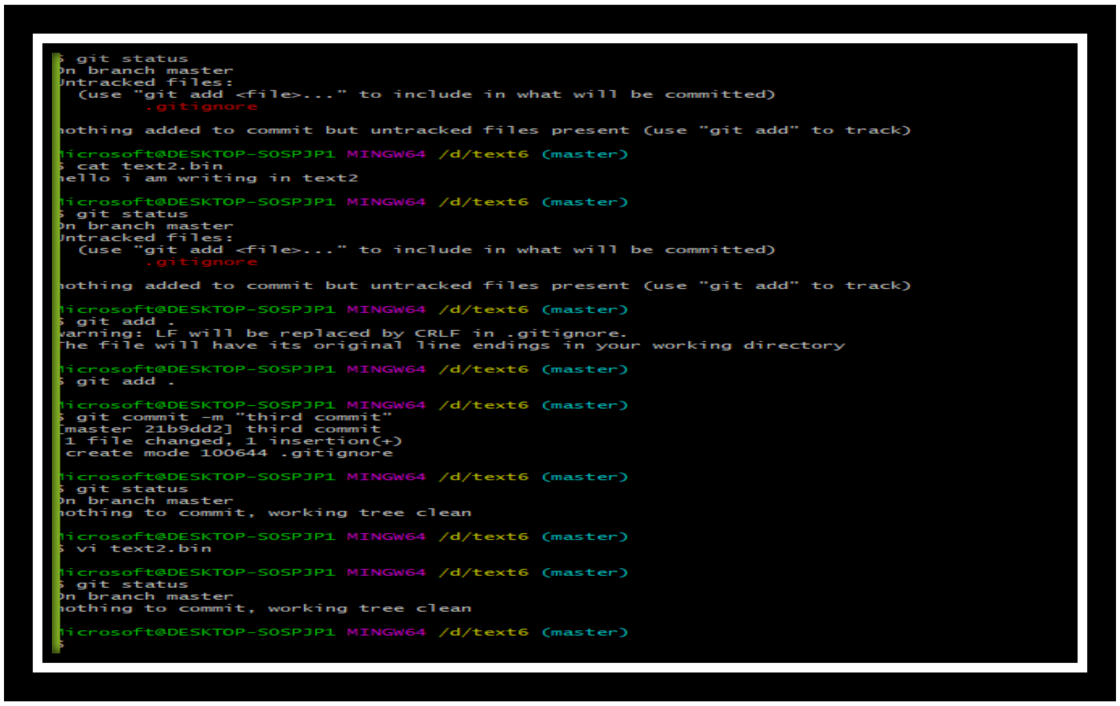
```
icrosoft@DESKTOP-SOSPJP1 MINGW64 ~
 cd D:

icrosoft@DESKTOP-SOSPJP1 MINGW64 /d
 mkdir text6

icrosoft@DESKTOP-SOSPJP1 MINGW64 /d
 cd text6

icrosoft@DESKTOP-SOSPJP1 MINGW64 /d/text6
 git init
nitialized empty Git repository in D:/text6/.git/

icrosoft@DESKTOP-SOSPJP1 MINGW64 /d/text6 (master)
 touch text1.txt

icrosoft@DESKTOP-SOSPJP1 MINGW64 /d/text6 (master)
 vi text6

icrosoft@DESKTOP-SOSPJP1 MINGW64 /d/text6 (master)
 vi text1

icrosoft@DESKTOP-SOSPJP1 MINGW64 /d/text6 (master)
 git add text1.txt

icrosoft@DESKTOP-SOSPJP1 MINGW64 /d/text6 (master)
 git add .
arning: LF will be replaced by CRLF in text1.
he file will have its original line endings in your working directory
arning: LF will be replaced by CRLF in text6.
he file will have its original line endings in your working directory

icrosoft@DESKTOP-SOSPJP1 MINGW64 /d/text6 (master)
 git add .

icrosoft@DESKTOP-SOSPJP1 MINGW64 /d/text6 (master)
 git commit -m "first commit"
master (root-commit) 7487caf] first commit
 3 files changed, 2 insertions(+)
 create mode 100644 text1
 create mode 100644 text1.txt
 create mode 100644 text6

icrosoft@DESKTOP-SOSPJP1 MINGW64 /d/text6 (master)
 git status
n branch master
othing to commit, working tree clean

icrosoft@DESKTOP-SOSPJP1 MINGW64 /d/text6 (master)
```



```
icrosoft@DESKTOP-SOSPJP1 MINGW64 /d/text6 (master)
 git reset --soft HEAD~1
atal: ambiguous argument 'HEAD~1': unknown revision or path not in the working tree.
se '--' to separate paths from revisions, like this:
git <command> [<revision>...] -- [<file>...]'

icrosoft@DESKTOP-SOSPJP1 MINGW64 /d/text6 (master)
 vi text1.txt

icrosoft@DESKTOP-SOSPJP1 MINGW64 /d/text6 (master)
 vi text1

icrosoft@DESKTOP-SOSPJP1 MINGW64 /d/text6 (master)
 git add .
arning: LF will be replaced by CRLF in text1.
he file will have its original line endings in your working directory

icrosoft@DESKTOP-SOSPJP1 MINGW64 /d/text6 (master)
 git add .

icrosoft@DESKTOP-SOSPJP1 MINGW64 /d/text6 (master)
 git commit -m "second commit"
master a6bab34] second commit
 1 file changed, 1 insertion(+)

icrosoft@DESKTOP-SOSPJP1 MINGW64 /d/text6 (master)
 git status
n branch master
othing to commit, working tree clean

icrosoft@DESKTOP-SOSPJP1 MINGW64 /d/text6 (master)
 git reset --soft HEAD~1

icrosoft@DESKTOP-SOSPJP1 MINGW64 /d/text6 (master)
 git status
n branch master
hanges to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   text1


icrosoft@DESKTOP-SOSPJP1 MINGW64 /d/text6 (master)
 notepad text1

icrosoft@DESKTOP-SOSPJP1 MINGW64 /d/text6 (master)
 git add .

icrosoft@DESKTOP-SOSPJP1 MINGW64 /d/text6 (master)
 git commit -m "second commit"
```



```
icrosoft@DESKTOP-SOSPJP1 MINGW64 /d/text6 (master)
$ git commit -m "second commit"
[master fc546b9] second commit
 1 file changed, 1 insertion(+)

icrosoft@DESKTOP-SOSPJP1 MINGW64 /d/text6 (master)
$ git reset --hard HEAD~1
HEAD is now at 7487caf first commit

icrosoft@DESKTOP-SOSPJP1 MINGW64 /d/text6 (master)
$ notepad text1

icrosoft@DESKTOP-SOSPJP1 MINGW64 /d/text6 (master)
$ touch text2.bin

icrosoft@DESKTOP-SOSPJP1 MINGW64 /d/text6 (master)
$ touch .gitignore

icrosoft@DESKTOP-SOSPJP1 MINGW64 /d/text6 (master)
$ vi .gitignore

icrosoft@DESKTOP-SOSPJP1 MINGW64 /d/text6 (master)
$ vi text2.bin

icrosoft@DESKTOP-SOSPJP1 MINGW64 /d/text6 (master)
$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .gitignore

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

icrosoft@DESKTOP-SOSPJP1 MINGW64 /d/text6 (master)
$ cat text2.bin
hello i am writing in text2

icrosoft@DESKTOP-SOSPJP1 MINGW64 /d/text6 (master)
$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .gitignore

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

icrosoft@DESKTOP-SOSPJP1 MINGW64 /d/text6 (master)
$ git add .
warning: LF will be replaced by CRLF in .gitignore.
```



```
$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .gitignore

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

icrosoft@DESKTOP-SOSPJP1 MINGW64 /d/text6 (master)
$ cat text2.bin
hello i am writing in text2

icrosoft@DESKTOP-SOSPJP1 MINGW64 /d/text6 (master)
$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .gitignore

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

icrosoft@DESKTOP-SOSPJP1 MINGW64 /d/text6 (master)
$ git add .
warning: LF will be replaced by CRLF in .gitignore.
The file will have its original line endings in your working directory

icrosoft@DESKTOP-SOSPJP1 MINGW64 /d/text6 (master)
$ git add .

icrosoft@DESKTOP-SOSPJP1 MINGW64 /d/text6 (master)
$ git commit -m "third commit"
[master 21b9dd2] third commit
 1 file changed, 1 insertion(+)
 create mode 100644 .gitignore

icrosoft@DESKTOP-SOSPJP1 MINGW64 /d/text6 (master)
$ git status
On branch master
nothing to commit, working tree clean

icrosoft@DESKTOP-SOSPJP1 MINGW64 /d/text6 (master)
$ vi text2.bin

icrosoft@DESKTOP-SOSPJP1 MINGW64 /d/text6 (master)
$ git status
On branch master
nothing to commit, working tree clean

icrosoft@DESKTOP-SOSPJP1 MINGW64 /d/text6 (master)
$
```