

AMITY UNIVERSITY

SOURCE CODE MANAGEMENT

CSE2002

LAB MANUAL

SUBMITTED BY: SUBMITTED TO:

Shreya NM to Monit Kapoor

A866175124051

B.TECH IN CSE(AIML) (2nd semester)

BATCH-07

SI. NO.	INDEX
---------	-------

- | | |
|---|---|
| • | Basics of Linux |
| • | Introduction-Git Bash |
| • | Git Bash And Git Hub |
| • | File Creation with commit and push commands |
| • | Branches Creation |

- Merge Request
- Open and Close Pull Request
- Complete Git Process

EXPERIMENT-01:

- Basics Of Linux:

1.What is Linux?

Linux is a family of free and open-source operating systems based on the Linux kernel. Operating systems based on Linux are known as Linux distributions or distros. Examples include Debian, Ubuntu, Fedora, CentOS, Gentoo, Arch Linux, and many others.

2. Linux Distributions (Distros)

Different versions of Linux tailored for various needs:

Ubuntu – User-friendly, great for beginners.

Debian – Stable and well-tested.

Fedora – Latest features, cutting-edge.

Arch Linux – Lightweight and customizable.

CentOS/RHEL – Used in enterprise environments.

3. Linux File System Structure

Linux follows a hierarchical structure:

bin → essential binary programs

etc → configuration files

home → user directories

root → root user's home directory

var variable data (logs, etc.)

tmp → temporary files

usr → user-installed software

4. Basic Linux Commands

Command Description

Is. – List files in a directory

Cd - Change directory

Pwd -Print working directory

Mkdir. Make a new directory

Rm - Remove files/directories

Cp. – Copy files/directories

Mv. - Move or rename files

Cat.- Display file content

Sudo. – Execute a command as superuser

man

Show manual/help for a command

5. File Permissions

Linux controls file access with permissions.

Use `ls -l` to view them (e.g., `-rwxr-xr--`).

Modify with `chmod`, `chown`, or `chgrp`.

6. Package Management

Install/update software using package managers:

Debian/Ubuntu: `apt` (e.g., `sudo apt install package-name`)

Red Hat/Fedora: `dnf` or `yum`

Arch Linux: `pacman`

7. Shell and Terminal

The shell interprets commands (e.g., `bash`, `zsh`).

Terminal is where you type the commands.

8. Users and Permissions

Regular users vs. Root (superuser).

Manage users with adduser, usermod, and passwd

Experiment-02:

Introduction -Git Bash

What Is Git Bash?

Git Bash is a command-line tool for Windows that provides:

Git command-line tools

A Bash (Unix-style) shell

. A way to run shell commands similar to those used in Linux/macOS

It's especially useful for developers using Git on Windows who want a Unix-like experience.

Key Features:

- Bash Emulation:
- Bash (Bourne Again Shell) lets you run Linux-style commands (e.g., ls, pwd, rm).

This makes it easier to follow tutorials written for Linux/macOS.

- Git Integration:
- You can run Git commands like git clone, git status, git commit directly in Git Bash.

Useful for version control and managing code repositories.

- Cross-Platform Compatibility:
- Lets Windows users interact with remote Linux servers or repositories more easily.

How to Install Git Bash:

Go to <https://git-scm.com>

Download the installer for your OS (choose Windows).

Run the installer and select “Git Bash” when prompted about default terminal.

After installation, right-click anywhere and choose “Git Bash Here” to open the terminal.

Common Commands in Git Bash:

Command.	Description
Ls.	List files and directories
Pwd	Print current directory path

Cd.	Change directory
Mkdir myfolder	Create a new folder
Rm filename	Remove a file
Git init	Initialize a new Git repositor
Command.	Description
Git clone URL.	Clone a repository from GitHub
Git status.	Show current Git status
Git add.	Stage all changes
Git commit -m “message” Commit staged changes	

Use Cases:

Managing Git repositories

Running shell scripts

Navigating your project with Unix-style commands

Automating development tasks

Experiment-03:

Git Bash And GitHub

What is Git?

Before connecting GitHub and Git Bash, it's important to know:

Git is a version control system to track changes in code.

GitHub is a hosting service for Git repositories.

- Git Bash lets you use Git on Windows with Linux-style

commands.

How They Work Together

Here's the typical workflow:

1. You install Git and Git Bash on your Windows machine.
2. You create or clone a Git repository using Git Bash.
3. You make changes to your files locally.
4. You use Git commands in Git Bash to:
 - Stage changes: `git add`
 - commit changes: `git commit`
 - push changes: `git push` (to GitHub)

5.You use GitHub to:

store your code in the cloud

collaborate with others

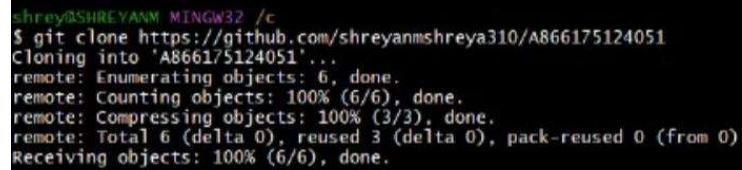
view project history, pull requests, issues, etc.

Example: Basic Workflow

Step-by-step using Git Bash & GitHub:

. 1.Clone a GitHub repo:

.

A terminal window with a black background and white text. The prompt is 'shrey@SHREYANM MINGW32 /c'. The command entered is '\$ git clone https://github.com/shreyanmshreya310/A866175124051'. The output shows the cloning process: 'Cloning into 'A866175124051'...', '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 3 (delta 0), pack-reused 0 (from 0)', and 'Receiving objects: 100% (6/6), done.'

```
shrey@SHREYANM MINGW32 /c
$ git clone https://github.com/shreyanmshreya310/A866175124051
Cloning into 'A866175124051'...
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 3 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (6/6), done.
```

```
shrey@SHREYANM MINGW32 /c
$ cd A866175124051
```

2.Make changes to our project files.

3.Stage and commit changes:

```
shrey@SHREYANM MINGW32 /c/A866175124051 (main)
$ git add .

shrey@SHREYANM MINGW32 /c/A866175124051 (main)
$ git commit -m "Add new feature"
On branch main
Your branch is up to date with 'origin/main'.
```

4.Push changes to GitHub:

```
shrey@SHREYANM MINGW32 /c/A866175124051 (main)
$ git push -u origin master
```

Experiment-04:

File Creation With Commit And Push

Command.

To Create a file,commit it and push it to a remote

Repository using Git Bash

1.Create a File.

Open Git Bash and create a folder

Using cd and mkdir create a folder:

```
shrey@SHREYANM MINGW32 /c/AB66175124051 (main)
$ cd c:

shrey@SHREYANM MINGW32 /c
$ mkdir folder
mkdir: cannot create directory 'folder': File exists

shrey@SHREYANM MINGW32 /c
$ mkdir folder
mkdir: cannot create directory 'folder': File exists

shrey@SHREYANM MINGW32 /c
$ cd folder
```

Then git init command to create a new Git repository

```
shrey@SHREYANM MINGW32 /c/folder (master)
$ git init
Reinitialized existing Git repository in C:/folder/.git/

shrey@SHREYANM MINGW32 /c/folder (master)
```

Create a new file using vi command in Linux is used to open

And edit files using the vi editor, a powerful text editor available on most Unix-based systems:

```
shrey@SHREYANM MINGW32 /c/folder (master)
$ vi file.txt
```

Now open the file and write:

Basic vi commands;

- Insert Mode:

O Press i to start editing.

- Save and Exit:

O Press ESC then type :wq to save and exit.

O Use :q! To exit without saving.

```
I am Shreya NM  
age 18 years old
```

file.txt[+] [unix] (05:29 01/01/1970)

-- INSERT --

```
I am Shreya NM  
age 18 years old
```

file.txt[+] [unix] (05:29 01/01/1970)
-- INSERT --

2.Add the File to Git:

O Stage the file for commit :

```
shrey@SHREYANM MINGW32 /c/folder (master)
$ git add file.txt
warning: in the working copy of 'file.txt', LF will be replaced
```

O To add all files in the directory:

```
shrey@SHREYANM MINGW32 /c/A866175124051 (main)
$ git add .
```

3.Commit the File:

O Commit the changes with a message.

```
shrey@SHREYANM MINGW32 /c/A866175124051 (main)
$ git commit -m "Add new feature"
On branch main
Your branch is up to date with 'origin/main'.
```

4.Add and push to remote repository:

Push the changes to Git Hub (assuming origin is the remote and
Main is the branch.)

```
shrey@SHREYANM MINGW32 /c/folder (master)
$ git remote add folder "https://github.com/shreyanmshreya310/A866175124051"

shrey@SHREYANM MINGW32 /c/folder (master)
$ git remote
folder

shrey@SHREYANM MINGW32 /c/folder (master)
$ git push -u folder master
```

Experiment-05:

Branches Creation

In Git branches allow developers to work on different

Features or fixes without affecting the main Codebase. Here's how we can create and manage branches.

O Check the branches:

Check the branches using git branch command:

```
shrey@SHREYANM MINGW32 /c/folder (master)
$ git branch
* master
```

To create new branch ;

```
shrey@SHREYANM MINGW32 /c/folder (master)
$ git branch 123
```

O Switching to a Branch:

To switch to the newly created branch:

```
shrey@SHREYANM MINGW32 /c/folder (master)
$ git checkout 123
Switched to branch '123'
```

Experiment-06:

Merge Request

O A Merge Request(MR) is a feature used in Gitbased platforms like GitLab to propose and reviews Changes before merging them into the main Branch.It is similar to a Pull Request(PR) in GitHub.

How Merge Request Works:

1. Create a Branch

A developer creates a separate branch from the main

Branch (often called main or master) to work on a specific

Feature, bug fix, or enhancement.

2. Make Changes

The developer writes code, commits changes to their Branch, and pushes the branch to the remote repository.

3. Open a Merge Request

In the Git platform (e.g., GitLab, GitHub):

- The developer opens a merge request from their Feature branch into the target branch (usually main).
- They add a description, possibly link related issues, And assign reviewers.
- Reviewers (team members or maintainers) review The code.
- Automated checks (e.g., tests, linters) are run via CI/CD pipelines.
- Feedback might be given; the author can make Additional commits to address it.

6. Merge

The MR is merged into the target branch:

- Options include merge, squash and merge, or

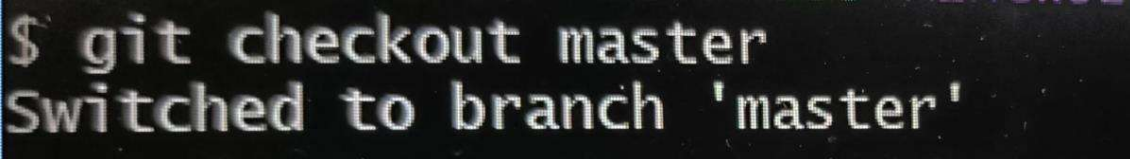
Rebase and merge.

- Once merged, the feature branch can be deleted if no

Longer needed.

Steps to Merge the Branch:

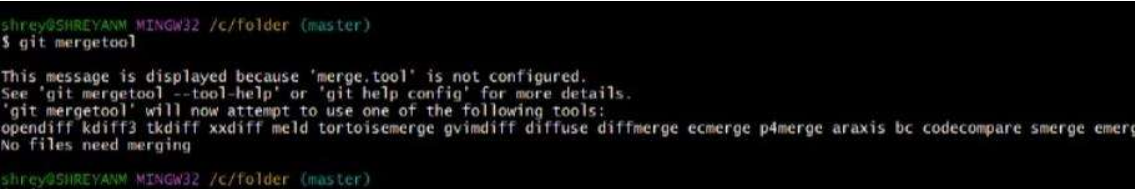
1. Switch to the main branch (master):



```
$ git checkout master
Switched to branch 'master'
```

2. Merge the branch:

Using git mergetool and git merge main command:



```
shrey@SHREYANM MINGW32 /c/folder (master)
$ 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 diffuze diffmerge ecmerge p4merge araxis bc codecompare smerge emerge
No files need merging

shrey@SHREYANM MINGW32 /c/folder (master)
```

```
shrey@SHREYANM MINGW32 /c/folder (master)
$ git merge test
merge: test - not something we can merge
```

Now Press enter to edit the files and windows are opening
,then remove some red lines and add some lines:



The screenshot shows a terminal window titled "MINGW32:/c/folder". Inside the terminal, the text "I am Shreya NM" and "age 18 years old" is displayed. Below this text, there are several lines of red characters, which appear to be a visual representation of a file's content or a list of items.

To remove merging commit one line:

```
shrey@SHREYANM MINGW32 /c/folder (master)
$ git commit -m "i written my name"
On branch master
```

Using the command graph we can see the graph of

Commits:

```
shrey@SHREYANM MINGW32 /c/folder (master)
$ git log --oneline --decorate --graph
* e85d3c8 (HEAD -> master, 123) i written my name
```

Benefits:

- Facilitates code review

- Triggers automated tests
- Maintains a clear change history
- Encourages collaborative development

Experiment-07:

• Open and Close Pull Request

1. Open a Pull Request

1. Push your changes to a branch on your fork or the

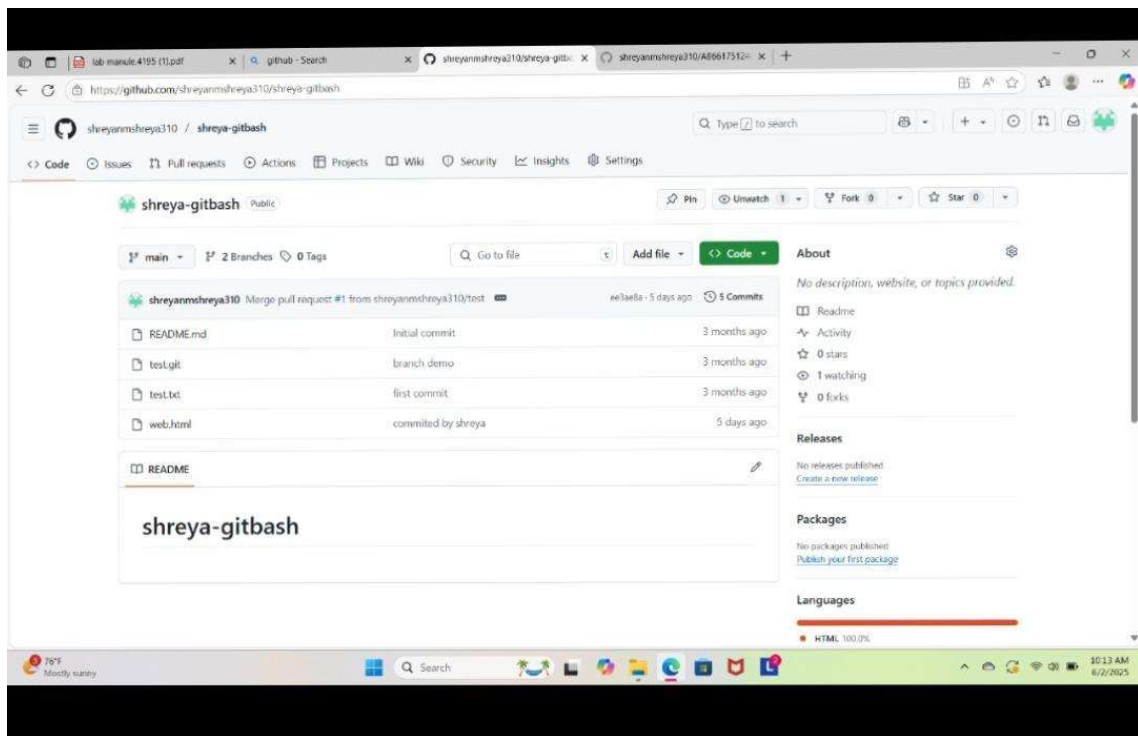
Same repository.

2. Go to GitHub, navigate to the repository.
3. You'll see a "Compare & pull request" button — click it.
4. Add a title and description for your PR.
5. Click "Create pull request".

2. Close a Pull Request

1. Click “Merge pull request”.
2. Confirm by clicking “Confirm merge”.
3. Optionally, delete the branch.

In the git hub account select the user which whom you
Want to merge and select the repo and fork it:



Now copy the link and using git clone command open that
File:

```
shrey@SHREYANM MINGW32 /c/123 (test)
$ git clone https://github.com/shreyanmshreya310/A866175124051
Cloning into 'A866175124051'...
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 3 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (6/6), done.
```

```
shrey@SHREYANM MINGW32 /c/123 (test)
$ cd A866175124051

shrey@SHREYANM MINGW32 /c/123/A866175124051 (main)
$ ls
README.md

shrey@SHREYANM MINGW32 /c/123/A866175124051 (main)
$ vi README.md

shrey@SHREYANM MINGW32 /c/123/A866175124051 (main)
$ git add .

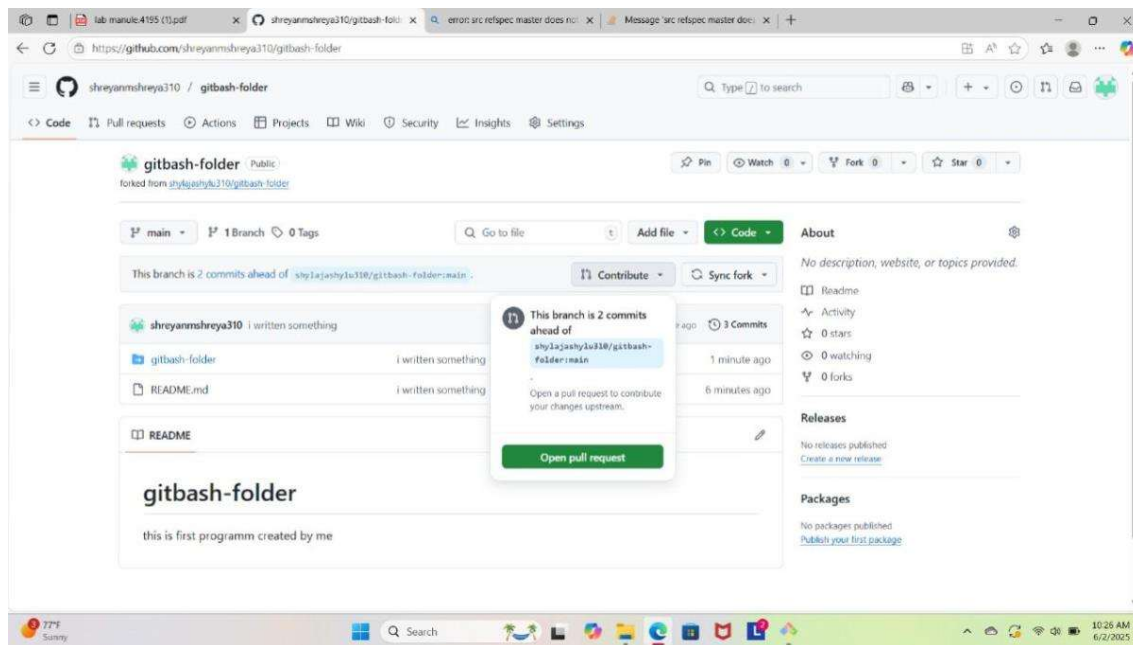
shrey@SHREYANM MINGW32 /c/123/A866175124051 (main)
$ git commit -m "i written something"
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean

shrey@SHREYANM MINGW32 /c/123/A866175124051 (main)
$ git remote
origin

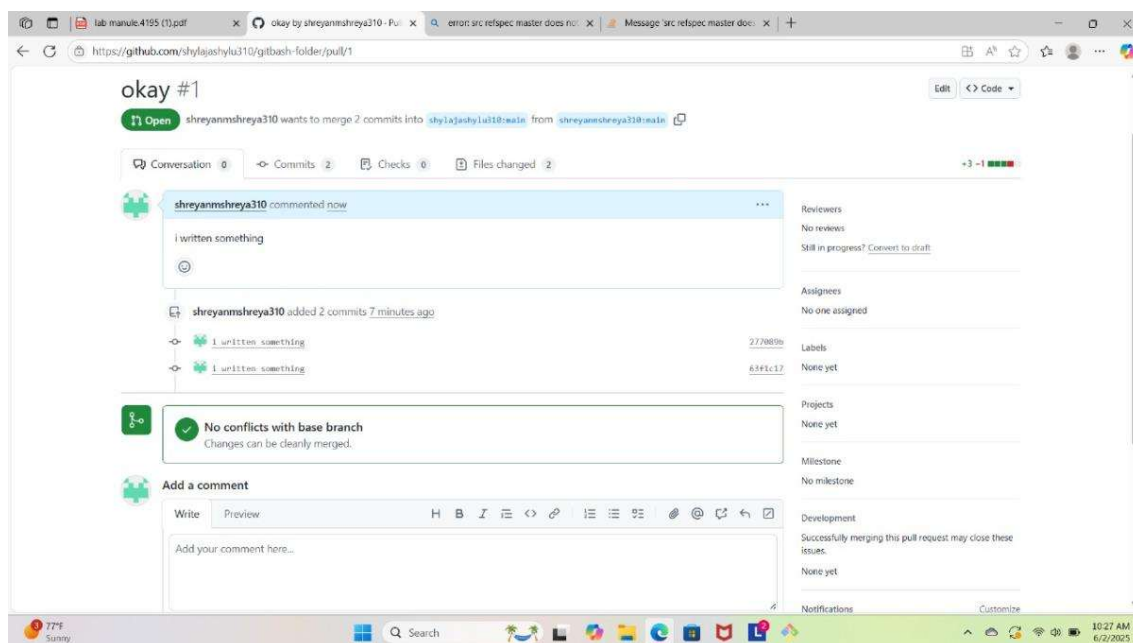
shrey@SHREYANM MINGW32 /c/123/A866175124051 (main)
$ git push -u origin master
```


In the Git Hub account contribute to open the pull request:



Steps to Close a Pull Request on GitHub:

1. Go to the repository on GitHub.
2. Click on the “Pull requests” tab
3. Find the pull request you want to close and click on it.
4. Scroll to the bottom of the PR page.
5. Click the “Close pull request” button.



Experiment 08:

- Complete Git Process.

1. Install Git Bash

- Download Git for Windows from Git's official

Website.

- Run the installer and follow the setup

Instructions.

- Choose Git Bash as the default terminal option.

2. Initialize a Repository

- Open Git Bash and navigation to our project folder.

```
shrey@SHREYANM MINGW32 /c/folder (master)
$ pwd
/c/folder

shrey@SHREYANM MINGW32 /c/folder (master)
$ ls
A866175124051/  file.txt

shrey@SHREYANM MINGW32 /c/folder (master)
$ cd c:

shrey@SHREYANM MINGW32 /c
$ mkdir 123

shrey@SHREYANM MINGW32 /c
$ cd 123
```

- Initialize a new git repository

```
shrey@SHREYANM MINGW32 /c/123
$ git init
Initialized empty Git repository in C:/123/.git/
```

3.Create and Modify Files

- Create a file using vi command:

```
shrey@SHREYANM MINGW32 /c/123 (master)
$ vi 123.txt
```

4. Check Repository Status

- View changes

```
shrey@SHREYANM MINGW32 /c/123 (master)
$ git status
On branch master
```

6. Stage and Commit Changes

- Add files to the staging area

```
shrey@SHREYANM MINGW32 /c/123 (master)
$ vi 765.txt
```

- Commit changes:

```
shrey@SHREYANM MINGW32 /c/123/A866175124051 (main)
$ git add .

shrey@SHREYANM MINGW32 /c/123/A866175124051 (main)
$ git commit -m "i written something"
On branch main
Your branch is up to date with 'origin/main'.
```

7. Create and Manage Branches

- Create a new branch:

```
shrey@SHREYANM MINGW32 /c/123 (master)
$ git branch
* master

shrey@SHREYANM MINGW32 /c/123 (master)
$ git branch test
```

- Switch to the branch:

```
shrey@SHREYANM MINGW32 /c/123 (master)
$ git checkout test
Switched to branch 'test'
```

```
shrey@SHREYANM MINGW32 /c/123 (test)
$ git remote add folder https://github.com/shreyanmshreya310/A866175124051

shrey@SHREYANM MINGW32 /c/123 (test)
$ git remote
folder
```

- Merge the branch into the main branch:

```
shrey@SHREYANM MINGW32 /c/123 (test)
$ git mergetool
```

```
shrey@SHREYANM MINGW32 /c/123 (test)
$ git merge master
```

```
shrey@SHREYANM MINGW32 /c/123 (test)
$ git log --oneline --decorate --graph
* 3bf2480 (HEAD -> test, master) i written something
```

8. Git Clone:

```
shrey@SHREYANM MINGW32 /c/123 (test)
$ git clone https://github.com/shreyanmshreya310/A866175124051
Cloning into 'A866175124051'...
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 3 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (6/6), done.
```

- Update local repository:

9. Pull Changes from Remote Repository

```
shrey@SHREYANM MINGW32 /c/123 (test)
$ git push -u origin master
```

10. Open and Close Pull Requests

- Open a Pull Request on GitHub and merge changes.
- Close the Pull Request if needed.

