AMITY UNIVERSITY BENGALURU

**SOURCE CODE MANAGEMENT**

******

***SUBMITTED BY: SUBMITTED TO:***

***AROHI D Dr. MONITH KAPOOR***

***A86605224182***

***B.TECH IN CSE***

**GIT INSTALLATION**

1. Install Git Bash

* + - * + Download Git for Windows from Git's official website.

1.Navigate to the [**official Git downloads page**](https://git-scm.com/download/win) and click the download link for the latest Git version for Windows:

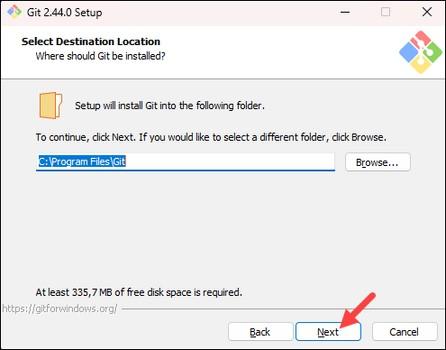


2. Double-click the downloaded [**file**](https://phoenixnap.com/glossary/what-is-a-file) to extract and launch the installer

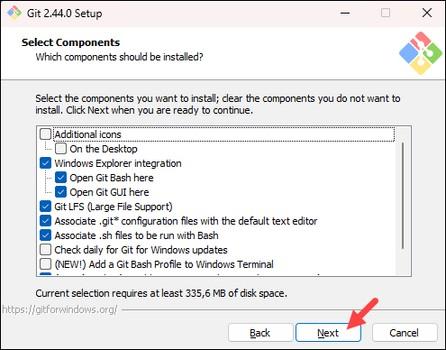
3.Review the [**GNU General Public License**](https://phoenixnap.com/glossary/gnu-general-public-license), and when you are ready to install, click **Next**.



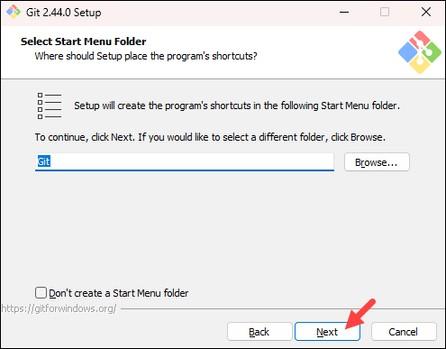
4.The installer prompts you for an installation location. Leave the default one unless you want to change it, and click **Next**.



5.In the component selection screen, leave the defaults unless you need to change them and click **Next**.

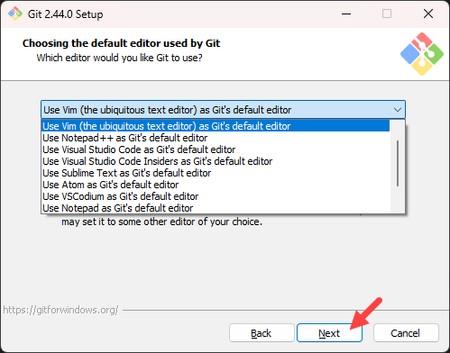


6.The installer offers to create a start menu [**folder**.](https://phoenixnap.com/glossary/what-is-a-folder) Click **Next** to accept and proceed to the next step.

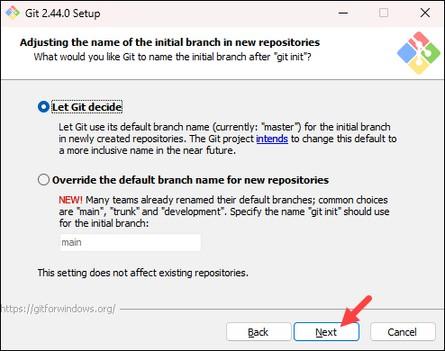


1. Select a text editor you want to use with Git. Use the drop-down menu to select Notepad++ (or whichever text editor you prefer) and click **Next**.

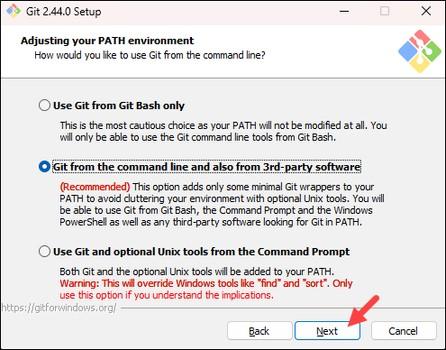
If you prefer to use a CLI text editor in [**Git Bash**,](https://phoenixnap.com/kb/what-is-git-bash) select [**nano**](https://phoenixnap.com/kb/use-nano-text-editor-commands-linux) or [**Vim**](https://phoenixnap.com/kb/vim-commands-cheat-sheet) from the list.



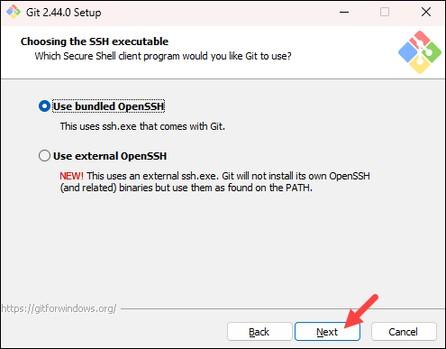
1. The next step allows you to choose a different name for your initial branch. The default is **master**. Unless you are working in a team that requires a different name, leave the default option and click **Next.**



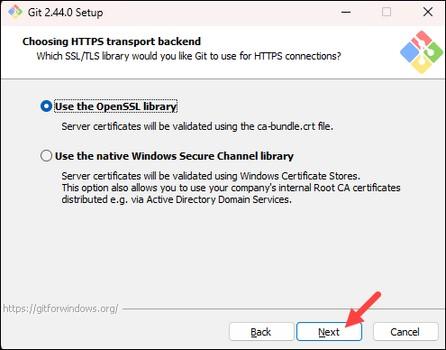
9.The next step allows you to change the **PATH environment**. The **PATH** is the default set of [**directories**](https://phoenixnap.com/glossary/what-is-a-directory) included when you run a command from the command line. Keep the middle (recommended) selection and click **Next**.



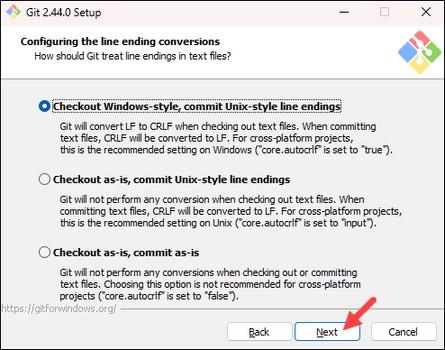
1. The installer prompts you to select the SSH client for Git to use. Git already comes with its own SSH client, so if you don't need a specific one, leave the default option and click **Next.**



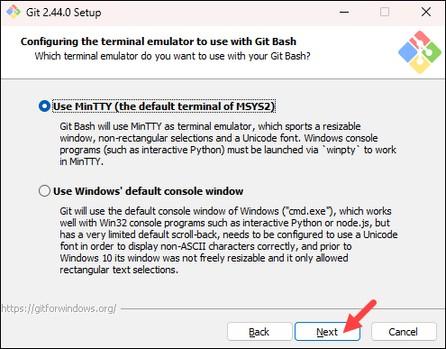
1. The next option relates to server certificates. The default option is recommended for most users. If you work in an Active Directory environment, you may need to switch to Windows Store certificates. Select your preferred option and click **Next**.



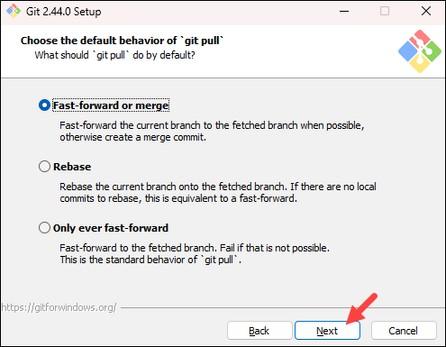
1. The following selection configures line-ending conversion, which relates to the way data is formatted. The default selection is recommended for Windows. Click **Next** to proceed.



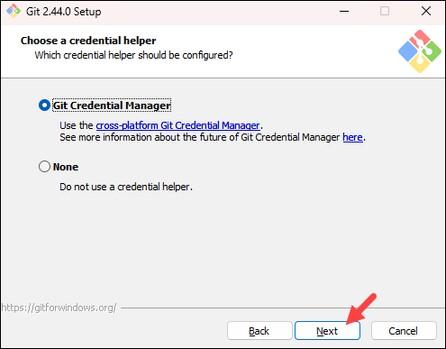
1. Choose the [**terminal emulator**](https://phoenixnap.com/glossary/terminal-emulation) you want to use. The default MinTTY is recommended for its features. Click **Next** to continue.



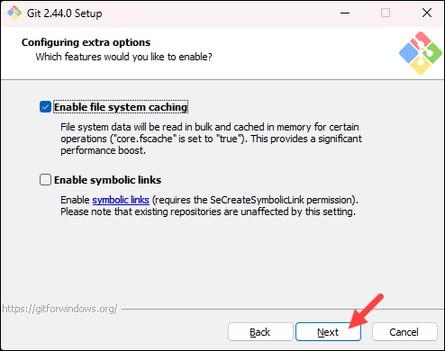
1. The next step allows you to choose what the **git pull** command will do. The default option is recommended unless you specifically need to change its behavior. Click **Next** to continue with the installation.



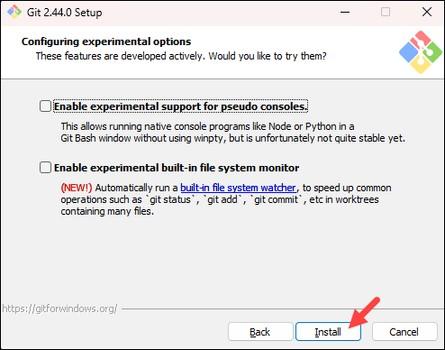
1. The next step is to choose which credential helper to use. Git uses credential helpers to fetch or save credentials. The default option is the most stable one. Select your preferred credential manager and click **Next**.



1. The next step lets you decide which extra options to enable. If you use [**symbolic links**](https://phoenixnap.com/kb/symbolic-link-linux), which represent shortcuts for the command line, tick the box. Keep [**file system**](https://phoenixnap.com/glossary/filesystem) caching checked and click **Next**.



1. Depending on which Git version you are installing, it may offer to install experimental features. At the time this article was written, the installer offered options to include support for pseudo controls and a built-in file system monitor. For the most stable operation, do not install experimental features and click **Install**.



1. Once the installation is complete, tick the boxes to view the Release Notes or launch Git Bash if you want to start using Git right away, and click **Finish**.



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.

1. **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.



1. **Linux File System Structure**

Linux follows a hierarchical structure:

* 1. bin → essential binary programs
  2. etc → configuration files
  3. home → user directories
  4. root → root user’s home directory
  5. var → variable data (logs, etc.)
  6. tmp → temporary files
  7. usr → user-installed software



1. **Basic Linux Commands**

**Command Description**

| ls | 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 |



1. **File Permissions** 
   * Linux controls file access with permissions.
   * Use ls -l to view them (e.g., -rwxr-xr--).
   * Modify with chmod, chown, or chgrp.



1. **Package Management** 
   * Install/update software using package managers:
     + - **Debian/Ubuntu**: apt (e.g., sudo apt install

package-name)

* + - * **Red Hat/Fedora**: dnf or yum o**Arch Linux**: pacman



1. **Shell and Terminal** 
   * The **shell** interprets commands (e.g., **bash**, **zsh**).
   * Terminal is where you type the commands.



1. **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**:**

1. Bash Emulation:

* + - * Bash (Bourne Again SHell) lets you run Linuxstyle commands (e.g., ls, pwd, rm).
      * This makes it easier to follow tutorials written for Linux/macOS.

2. Git Integration:

oYou can run Git commands like git clone, git status, git commit directly in Git Bash. oUseful for version control and managing code repositories.

3. Cross-Platform Compatibility:

oLets Windows users interact with remote Linux servers or repositories more easily.

Common Commands in Git Bash**:**

| Command | Description | |
| --- | --- | --- |
| ls | List files and directories | |
| pwd | Print current directory path | |
| Command | Description |
|  |  |
| cd | Change directory |
| mkdir myfolder | Create a new folder |
| rm filename | Remove a file |
| git init | Initialize a new Git repository |
| 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.



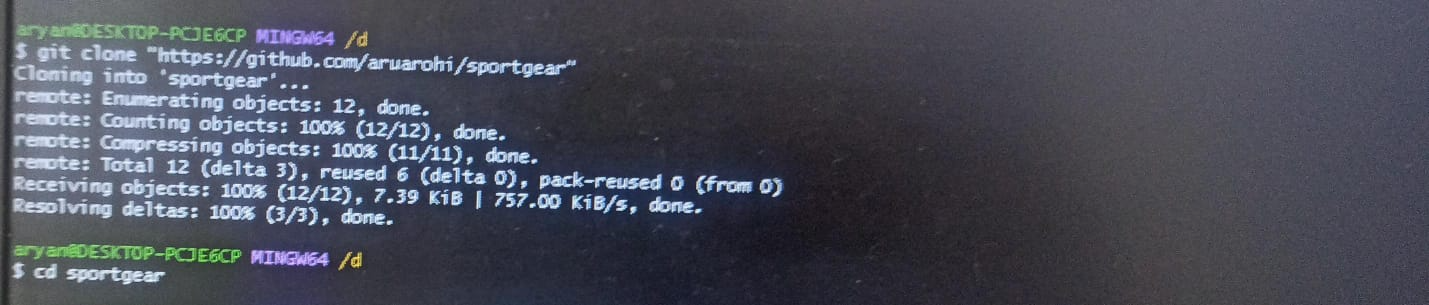
🔗 How They Work Together Here’s the typical workflow:

* 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: ostage changes: git add ocommit changes: git commit opush changes: git push (to GitHub)
  5. You use GitHub to:

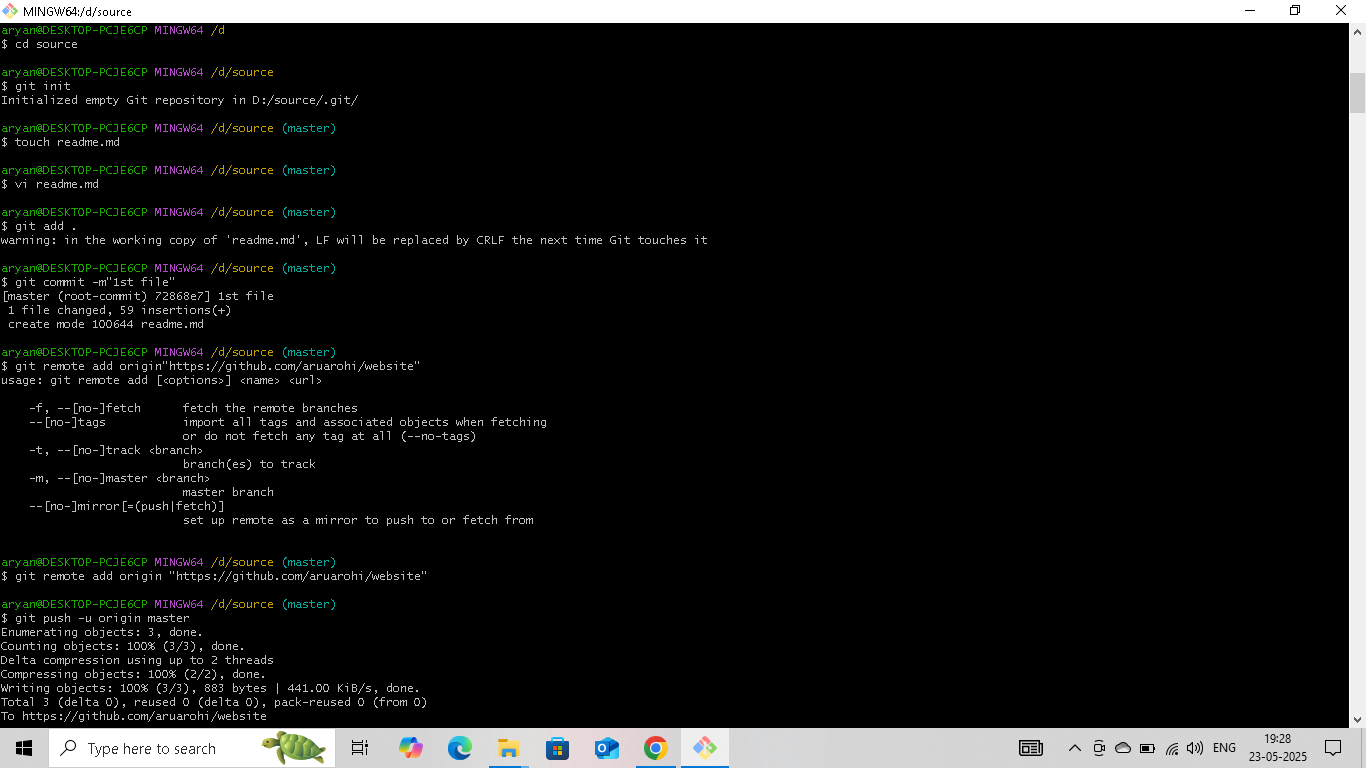
ostore your code in the cloud ocollaborate with others oview project history, pull requests, issues, etc**.**

Example: Step-by-step using Git Bash

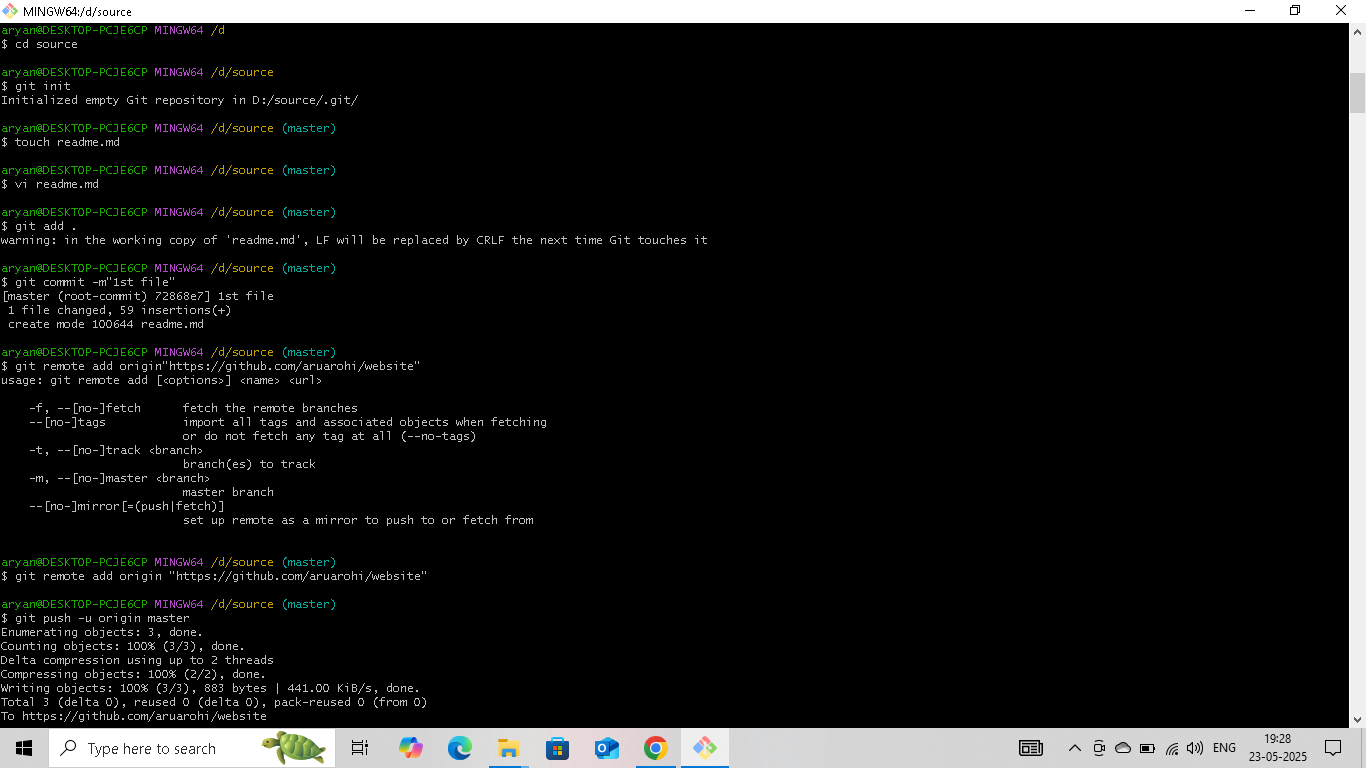
* 1. Clone a GitHub repo:



* 1. Make changes to our project files.
  2. Stage and commit changes:



4.Push changes to GitHub:



Experiment-04:

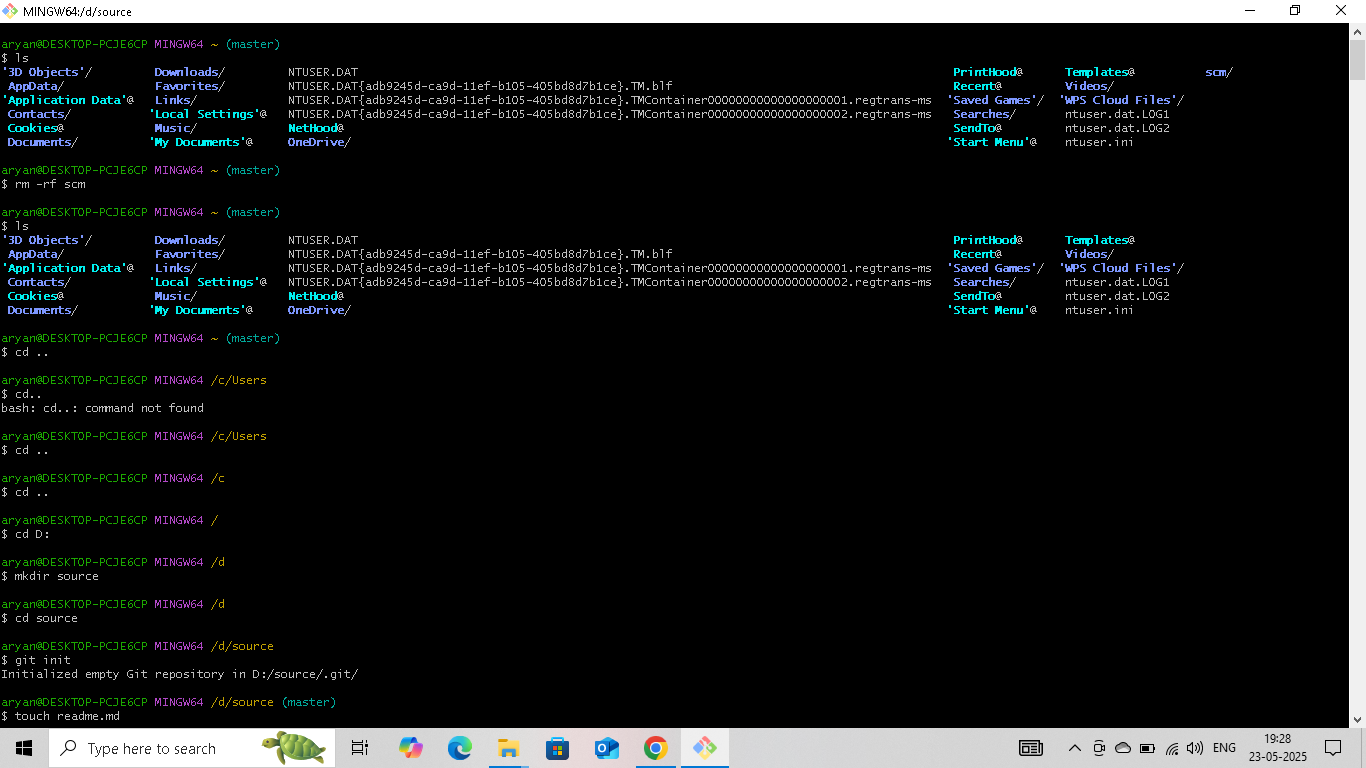
 **File Creation With Commit And Push Command.**

⮚ To Create a file,commit it and push it to a remote repository using Git Bash

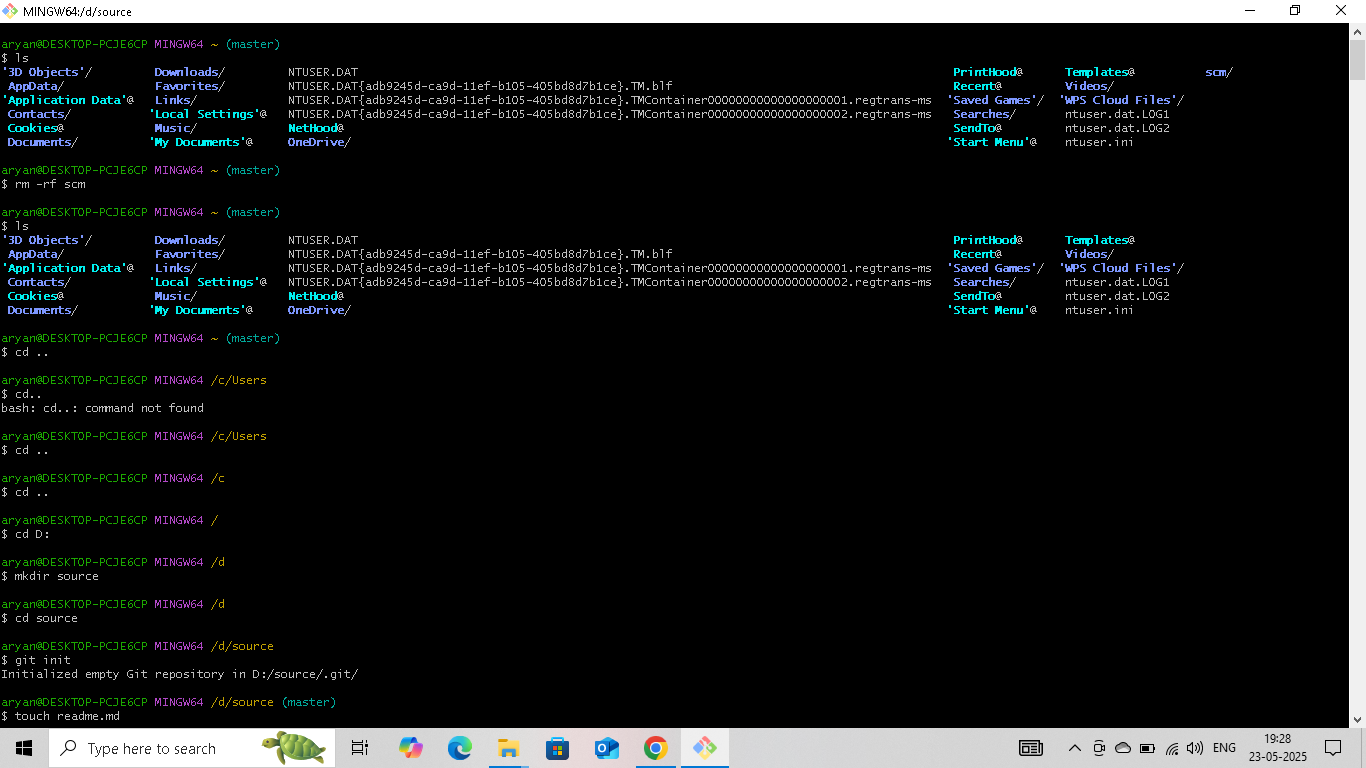
1.Create a File.

Open Git Bash and create a folder :

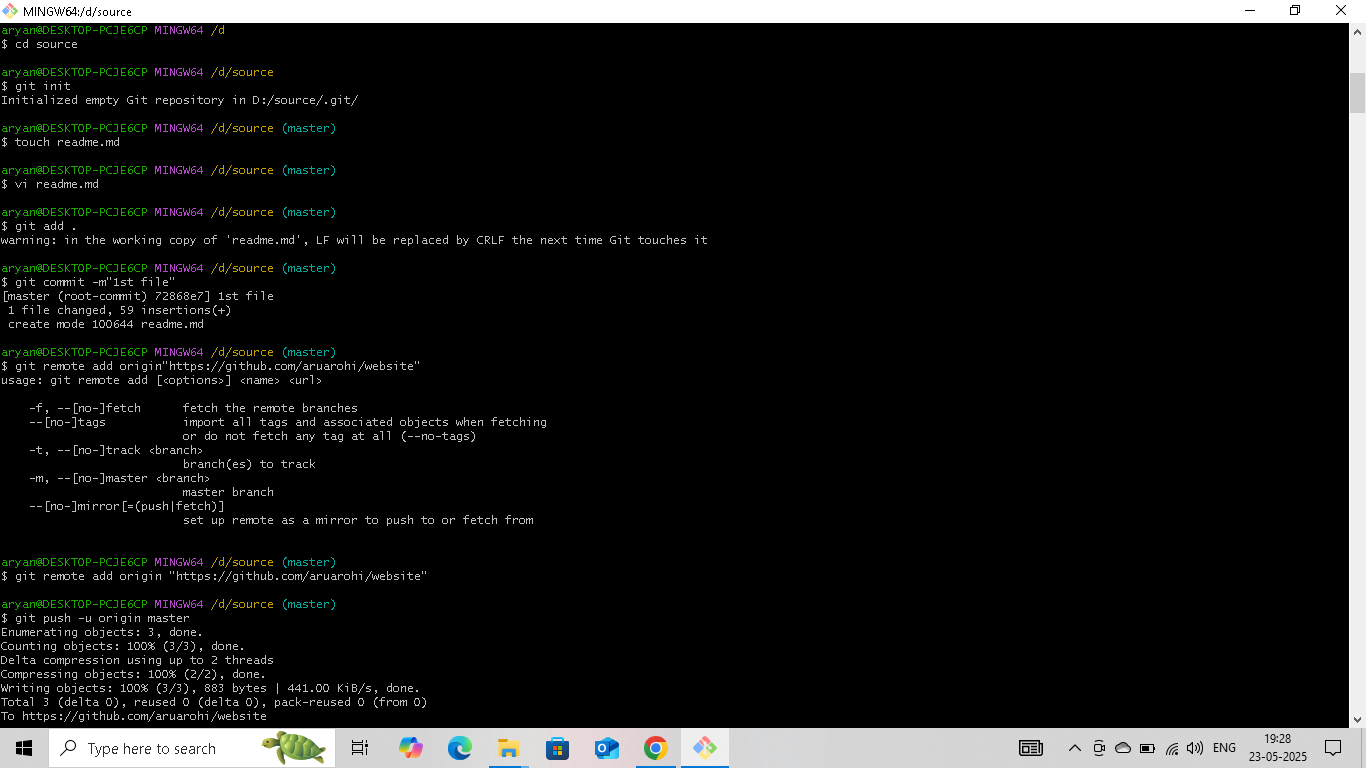
Using cd and mkdir create a folder:



Then git init command to create a new Git repository



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:



Now open the file and write:

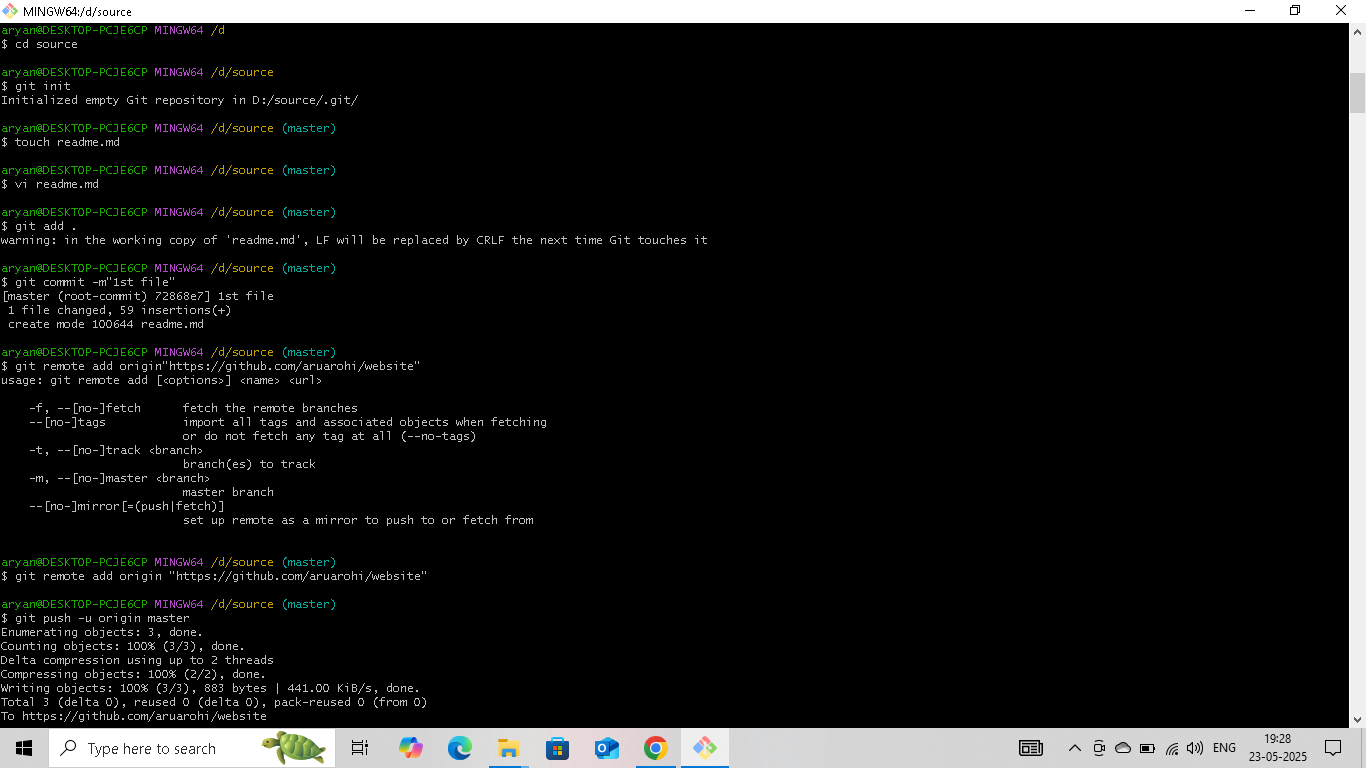
Basic vi commands;

* + Insert Mode: o Press **i** to start editing. 
  + Save and Exit:
    - * Press ESC then type :wq to save and exit. o Use :q! to exit without saving.

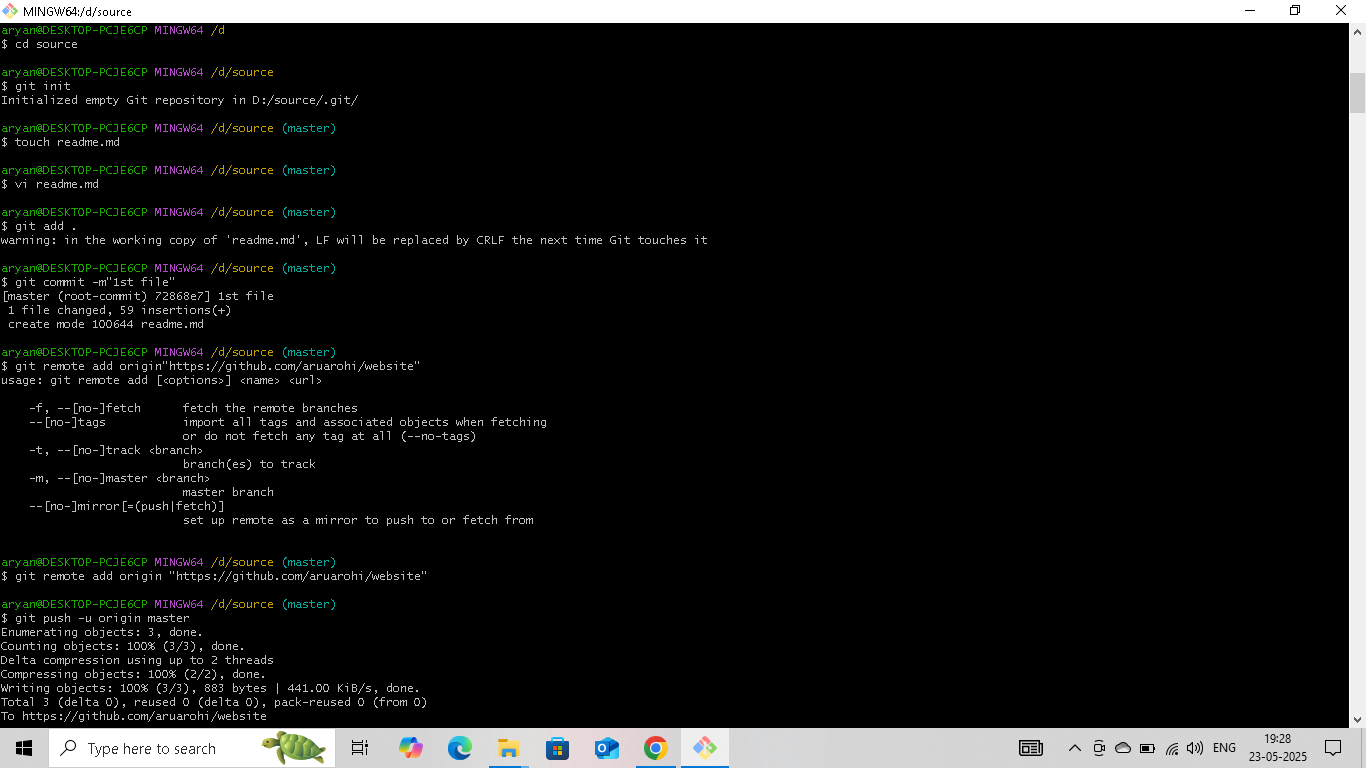


2.Add the File to Git: o Stage the file for commit :

* + - * To add all files in the directory:



3.Commit the File: o Commit the changes with a message.



4.Add and push to remote repository:

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

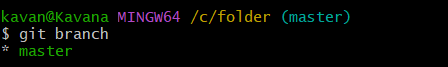
Experiment-05:

 **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.

* + - Check the branches:

Check the branches using git branch command:



To create new branch ;

* Switching to a Branch:

To switch to the newly created branch:



Experiment-06:

 **Merge Request**

o A Merge Request(MR) is a feature used in Gitbased platforms like GitLab to propose and review 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.

**4. Code Review**

* **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.

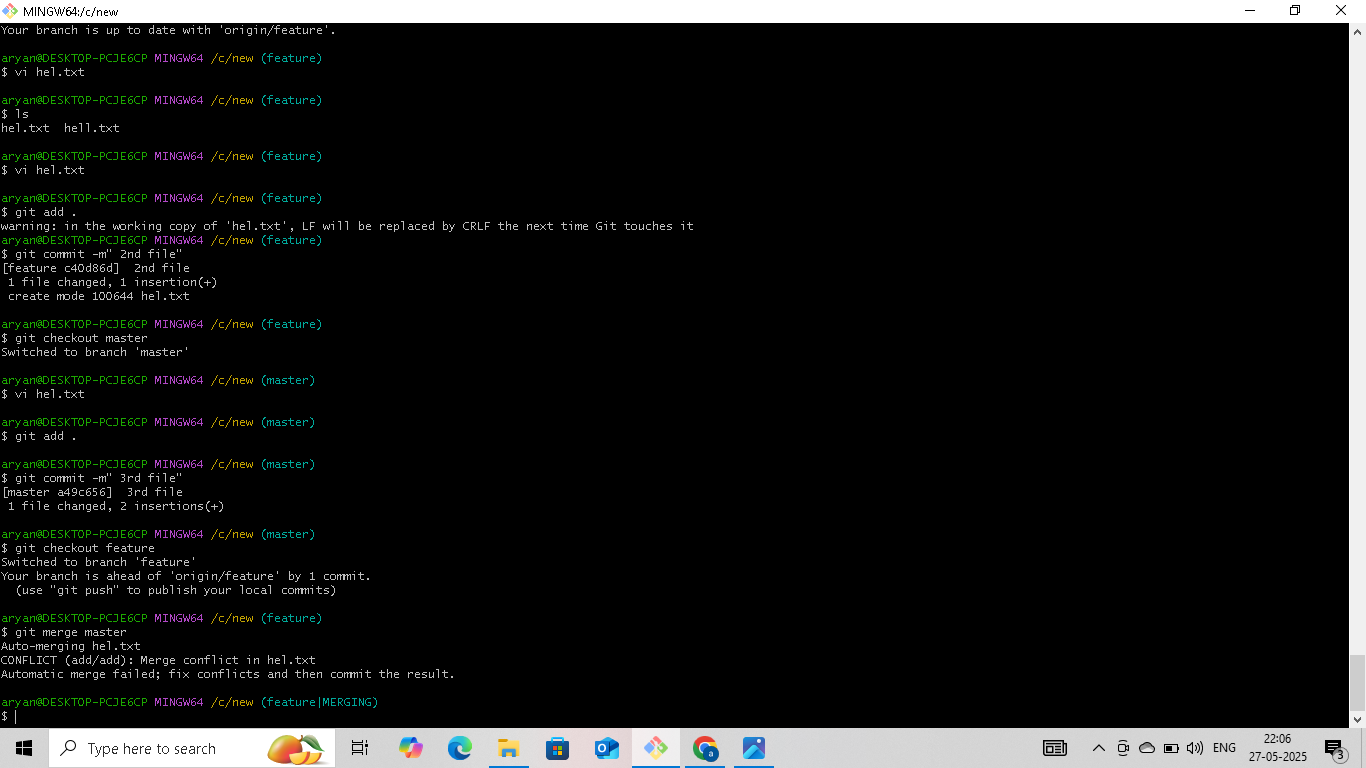
1. **Merge**

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.

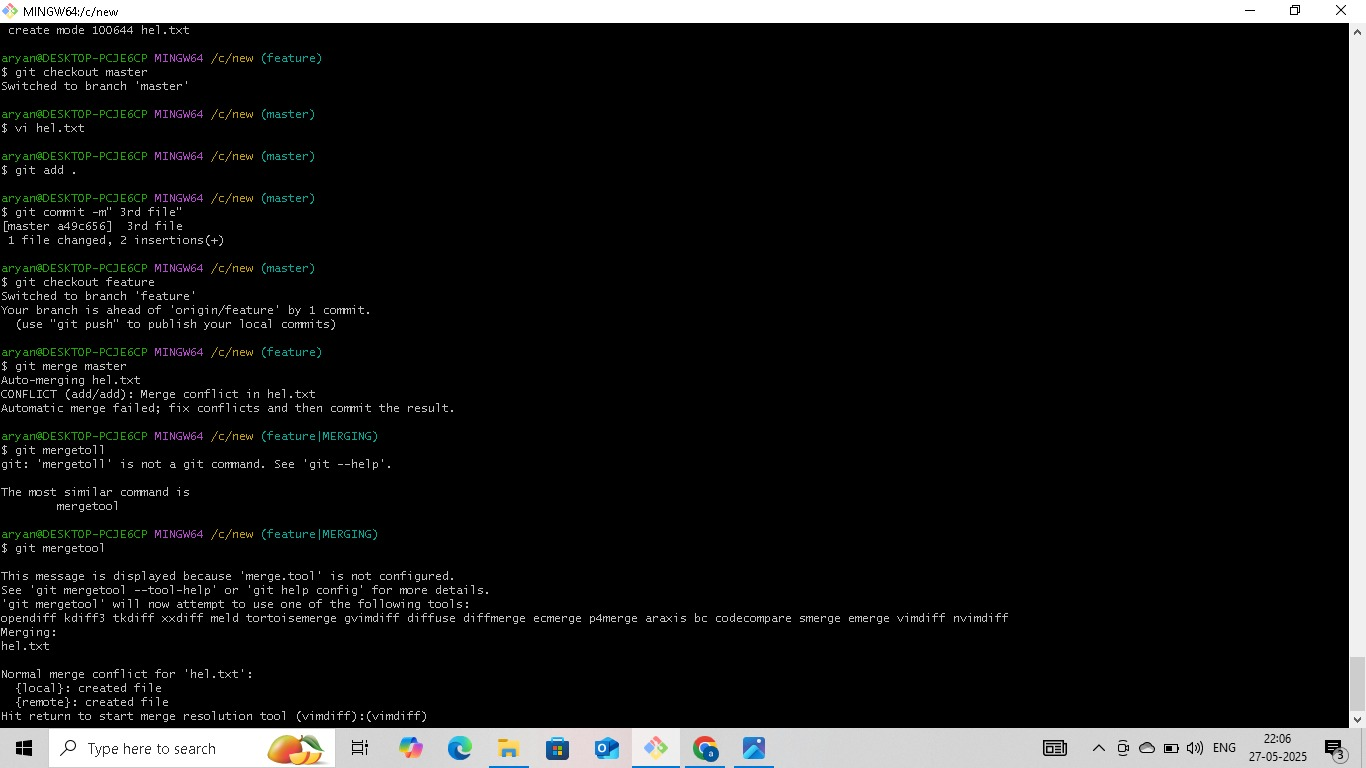
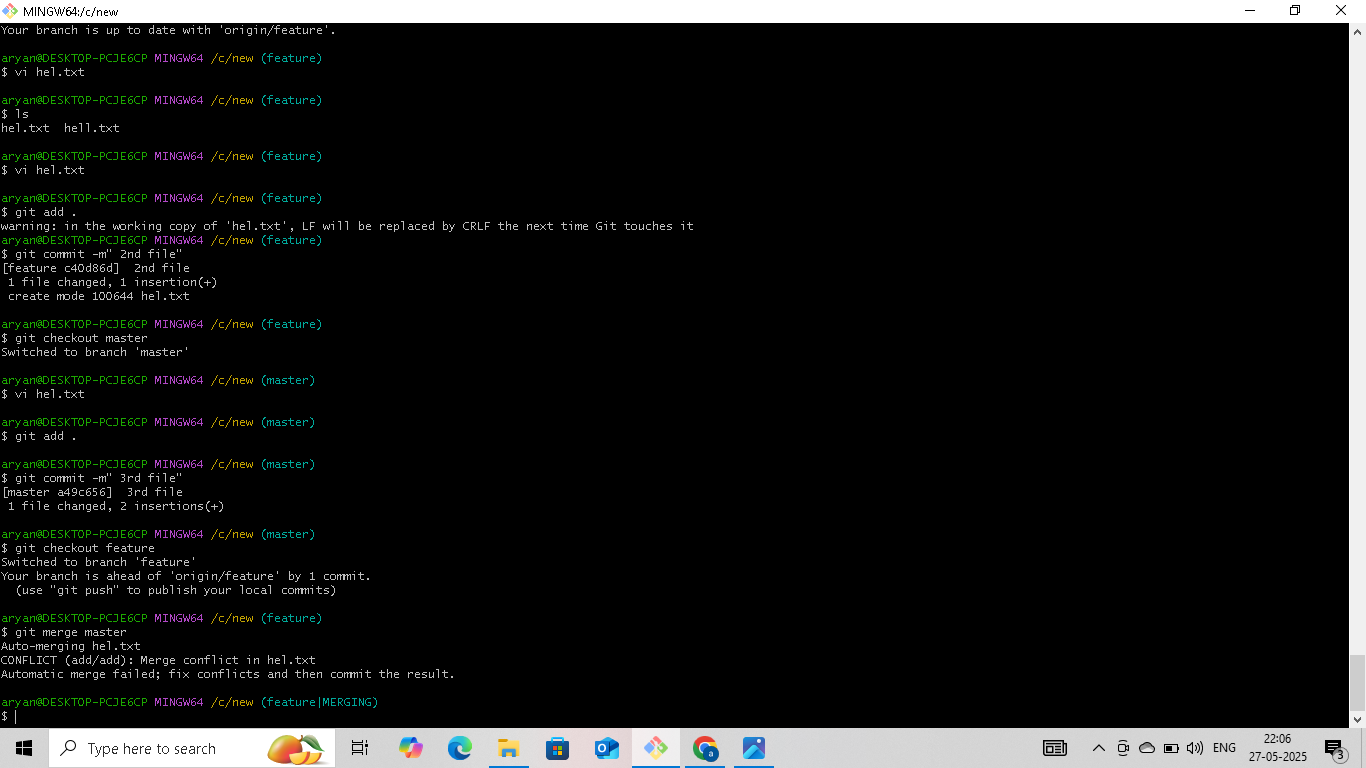
Steps to Merge the Branch:

1.Switch to the branch (feature):



2.Merge the branch:

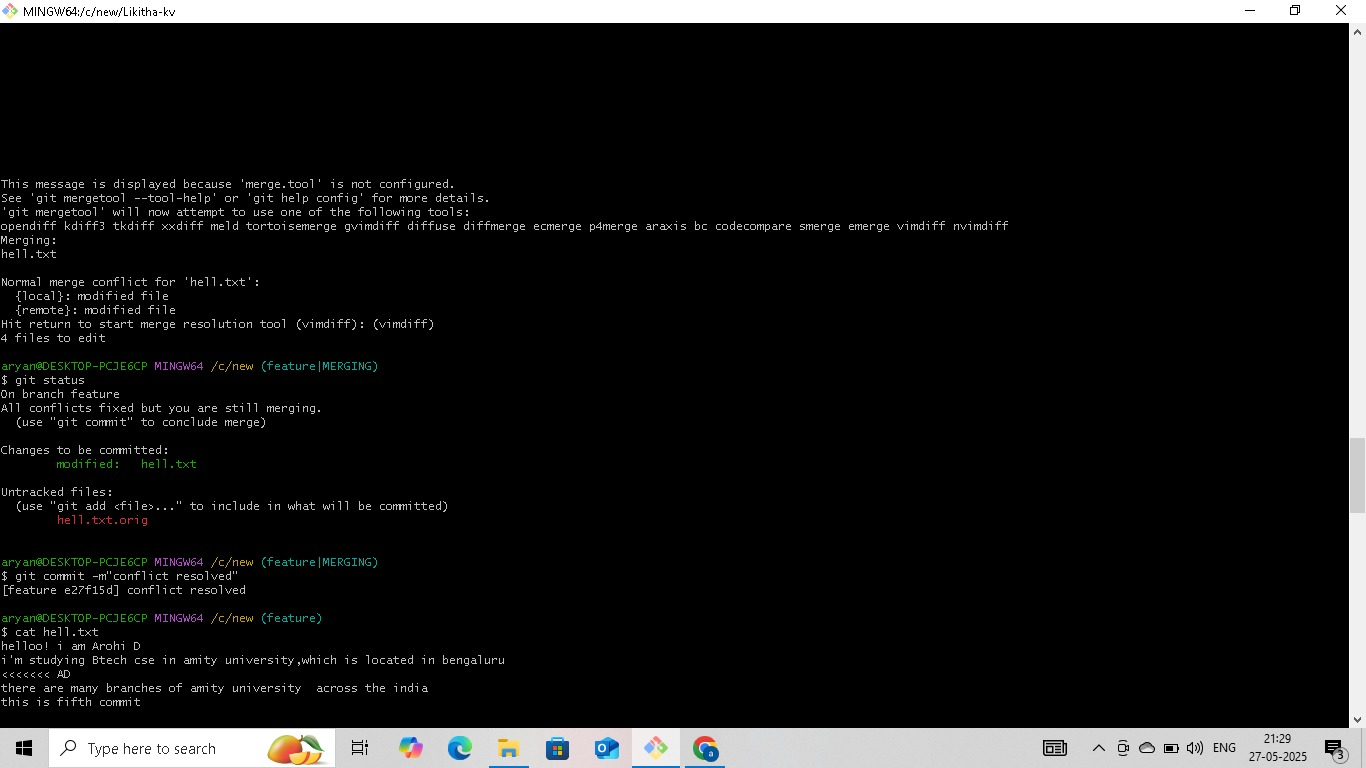
Using git mergetool and git merge main command:



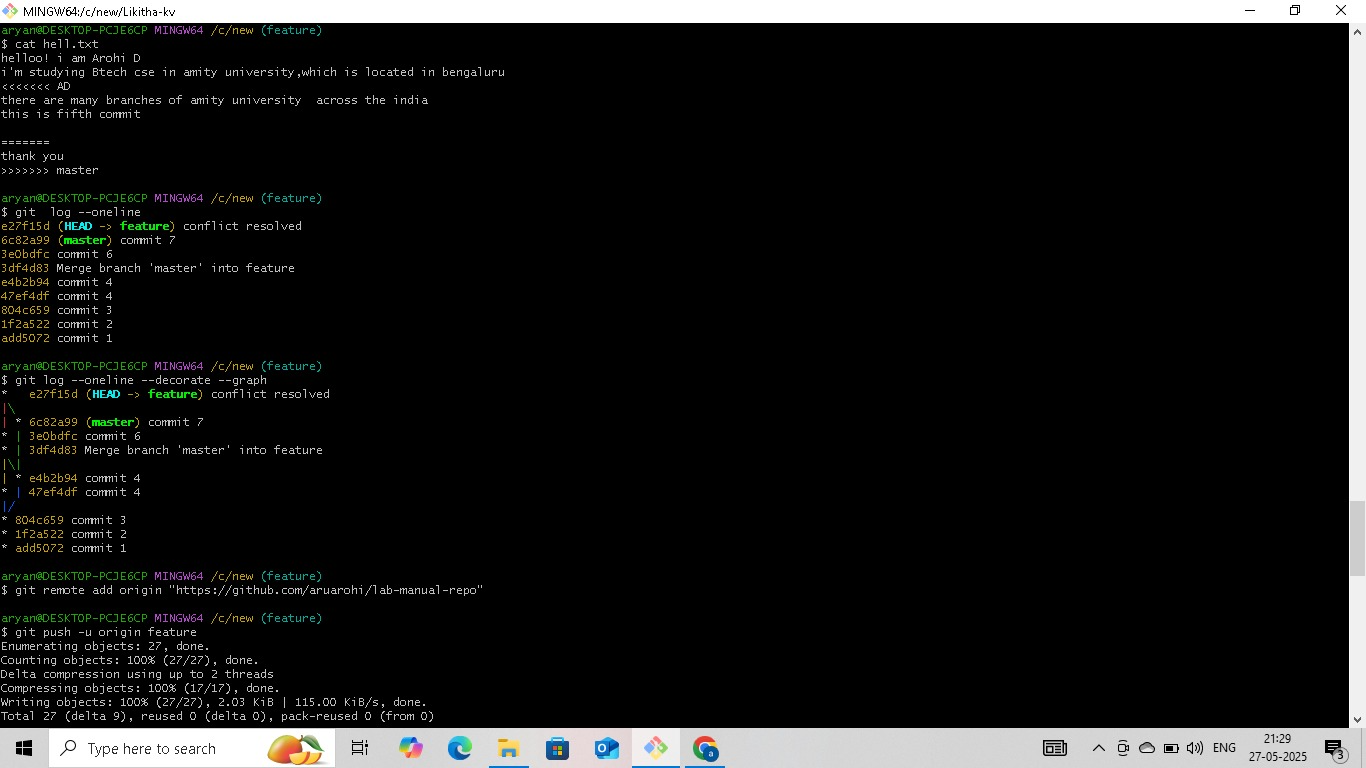
Now Press enter to edit the files and windows are opening ,then remove some red lines and add some lines:



To remove merging commit one line:



Using the command graph we can see the graph of commits:



**Benefits:**

* + - Facilitates **code review**
    - Triggers **automated tests**
    - Maintains a clear **change history**
    - Encourages **collaborative development**

Experiment-07:

* + - **Open and Close Pull Request**

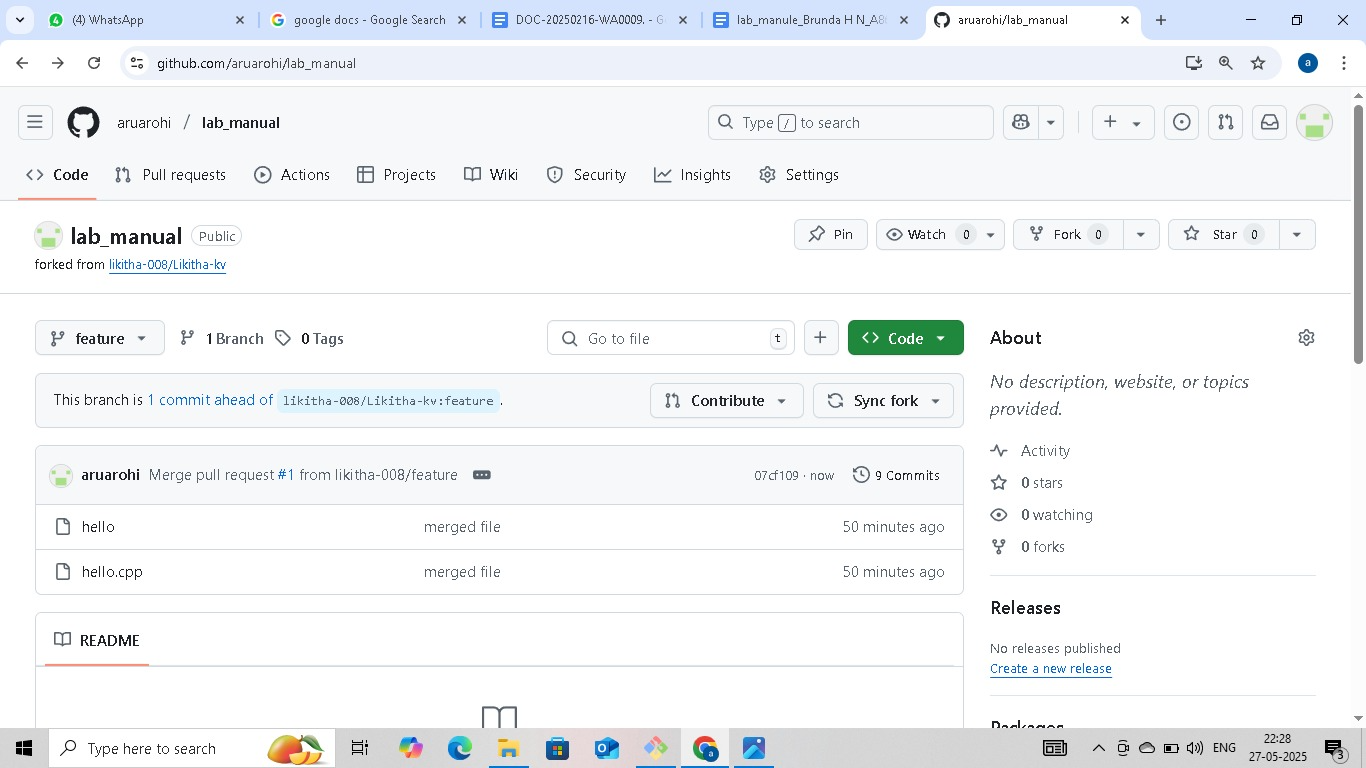
**1. Open a Pull Request**

* + 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”.

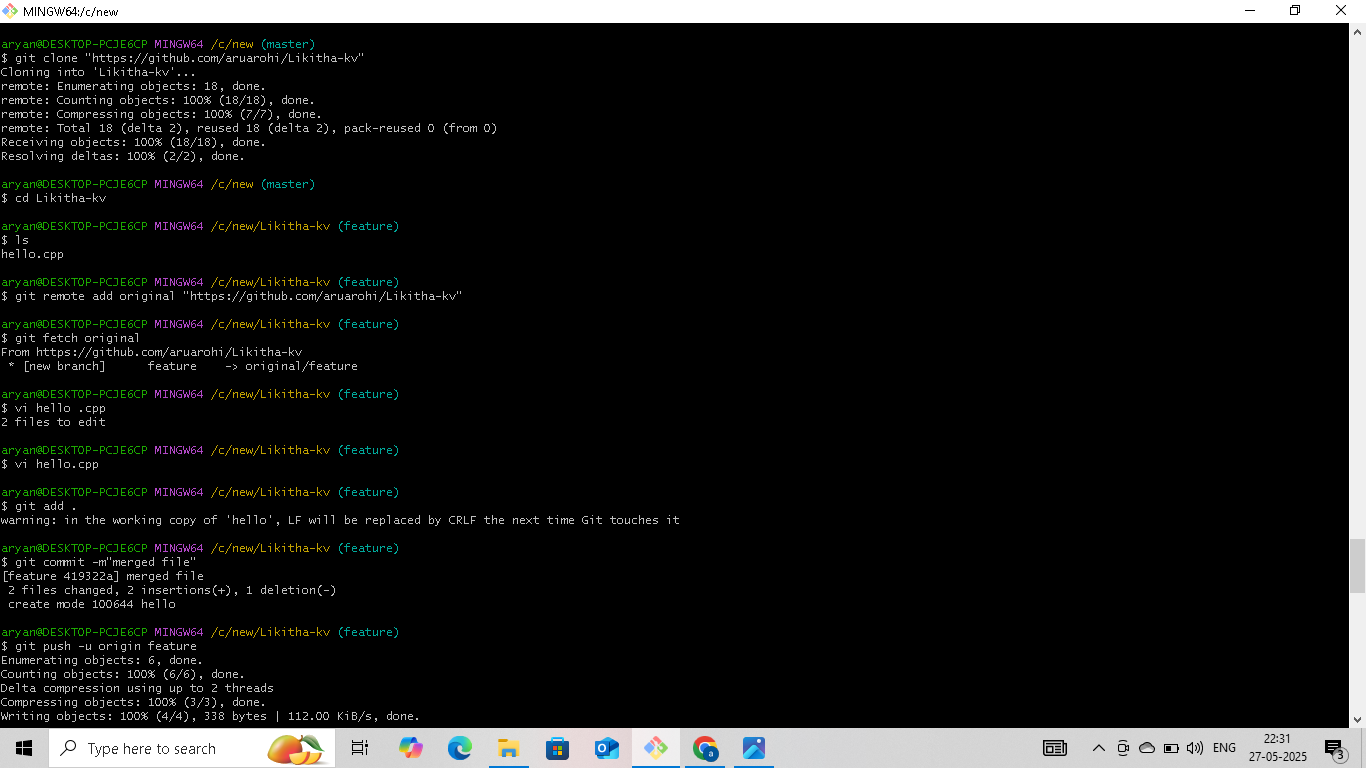
**2. Close a Pull Request**

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

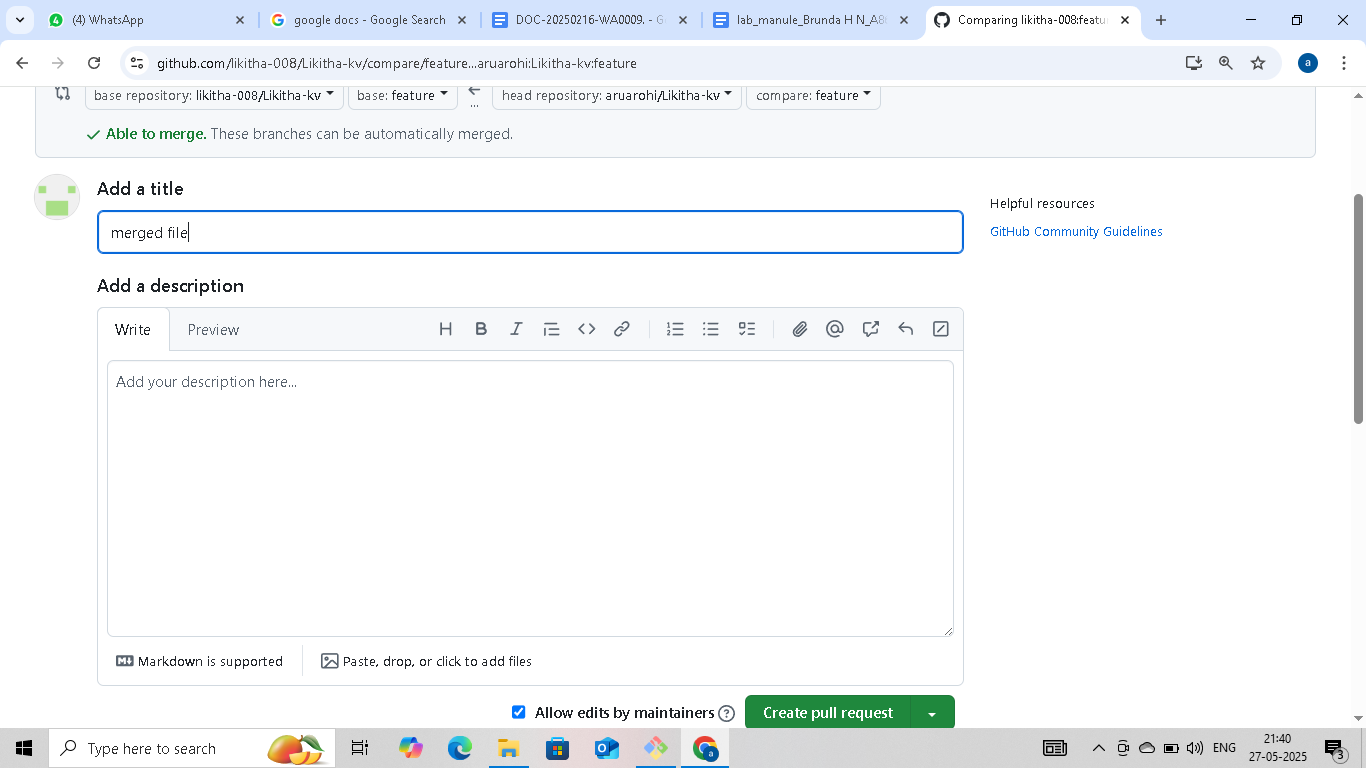
In the git hub account select the user which whom you want to merge and select the repo and fork it:



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



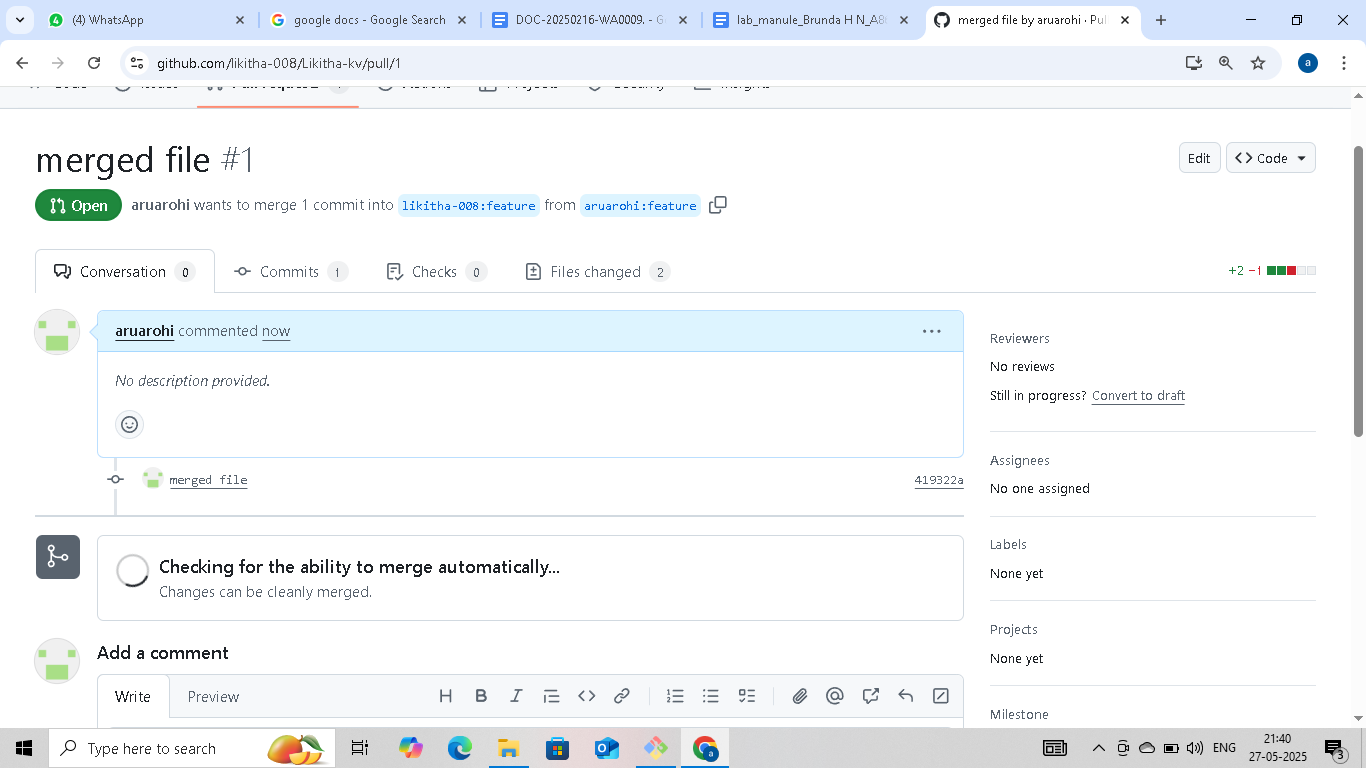
In the Git Hub account contribute to open the pull request:

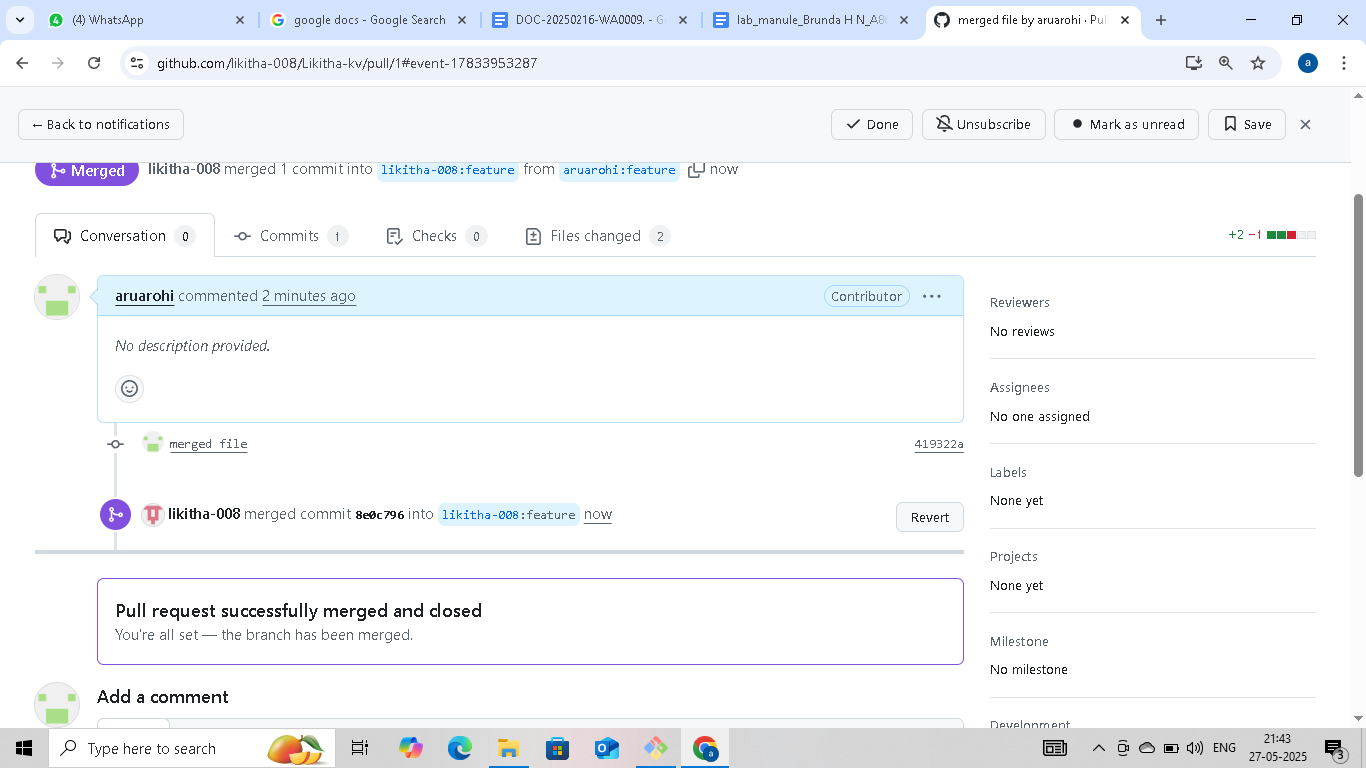


**Steps to Close a Pull Request on GitHub:**

1. **Go to the repository** on GitHub. 2. Click on the **"Pull requests"** tab.

* 1. Find the pull request you want to close and click on it.
  2. Scroll to the bottom of the PR page.
  3. Click the **“Close pull request”** button.





Experiment-08:

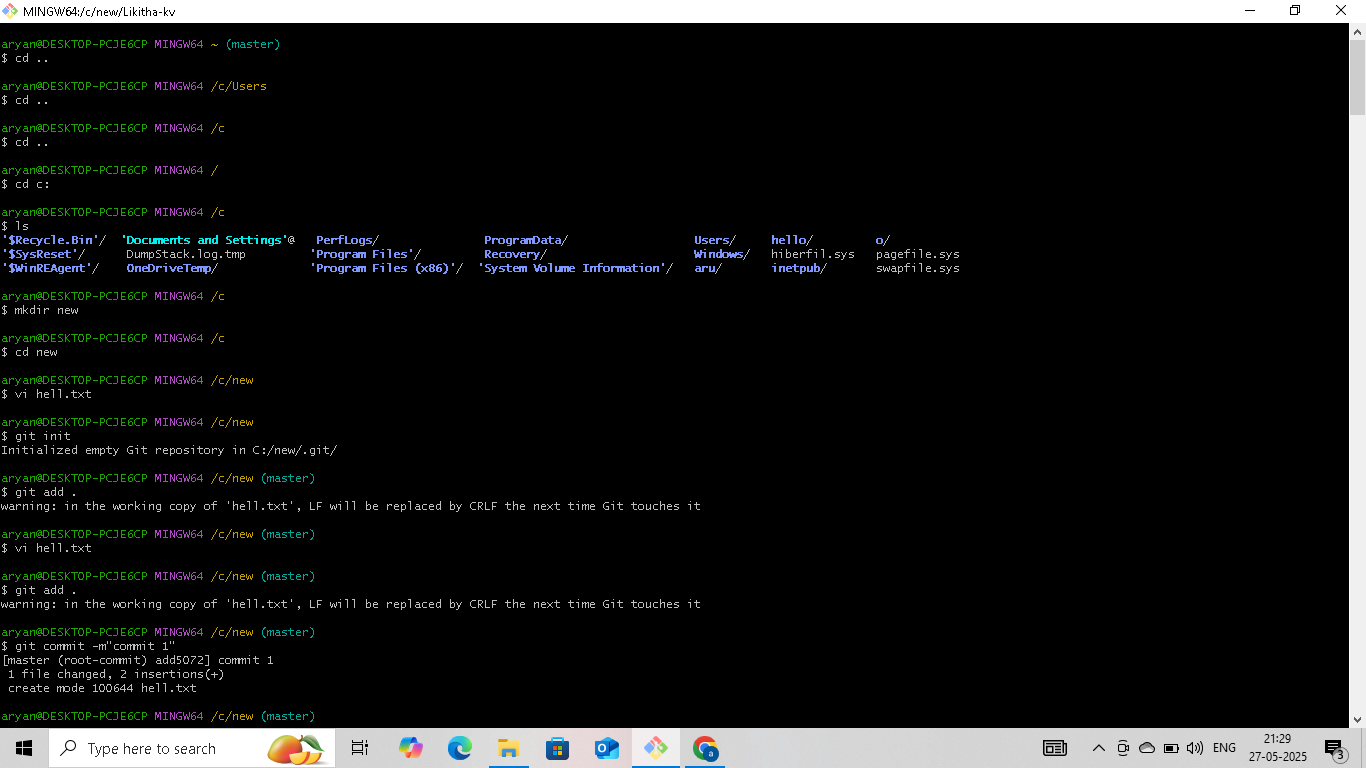
 **Complete Git Process.**

1. Install Git Bash

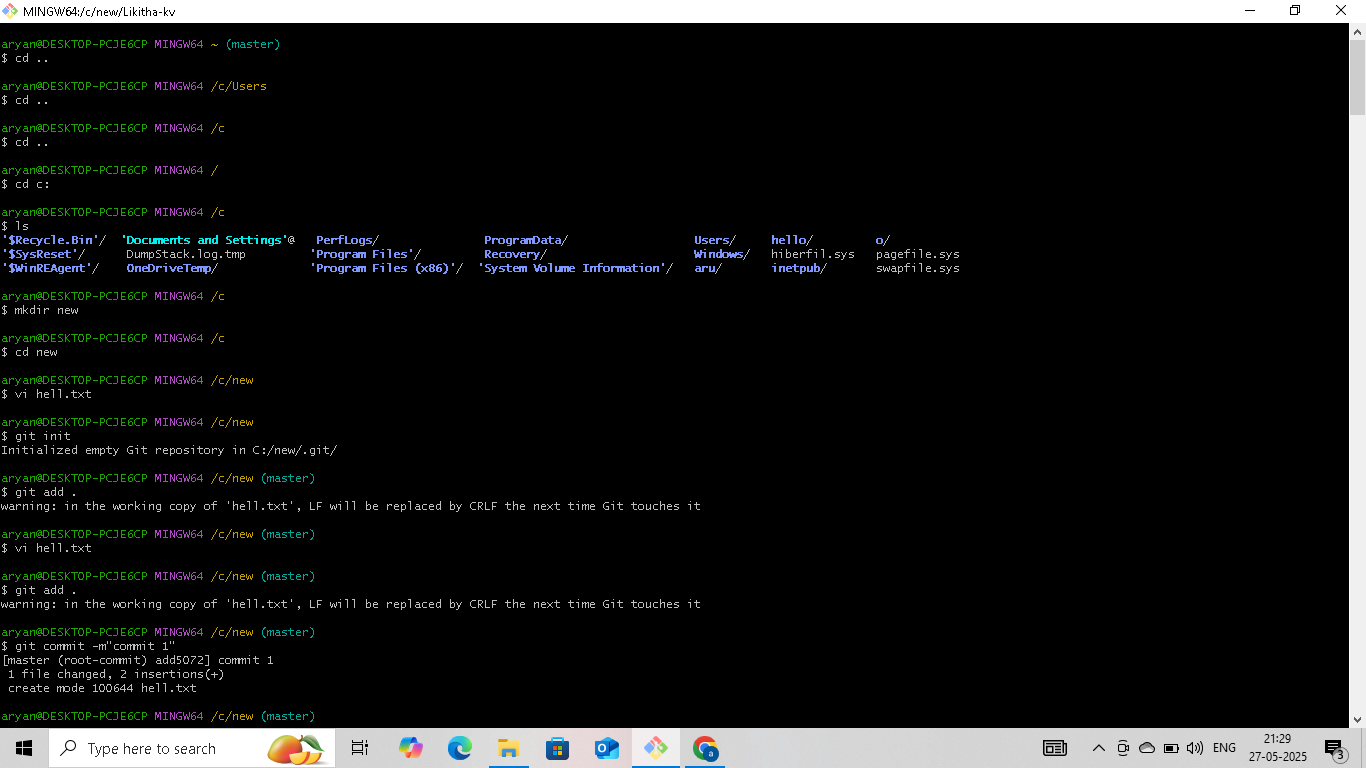
* + - * + Download Git for Windows from Git's official website.
        + Run the installer and follow the setup instructions.
        + Choose Git Bash as the default terminal option.

2.Initialize a Repository

* + - Open Git Bash and navigation to our project folder.

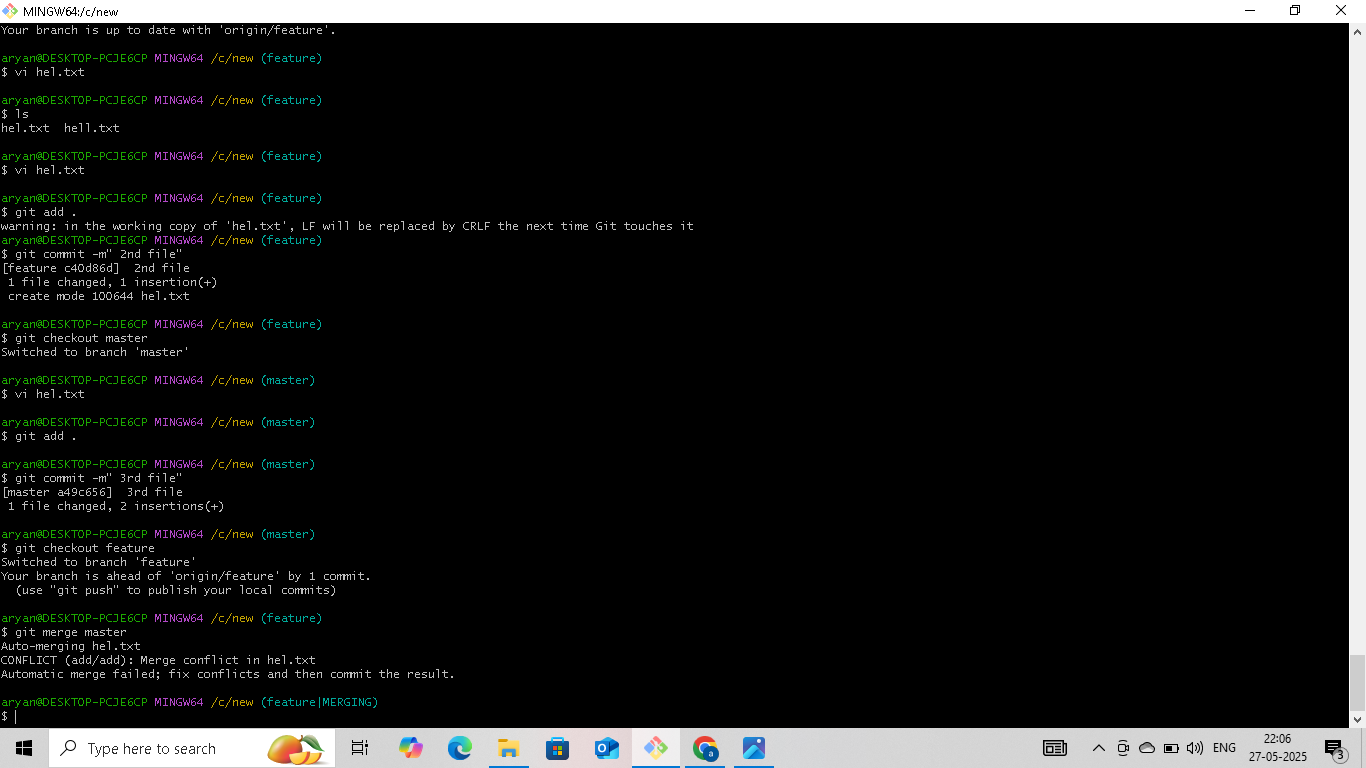


* + - Initialize a new git repository



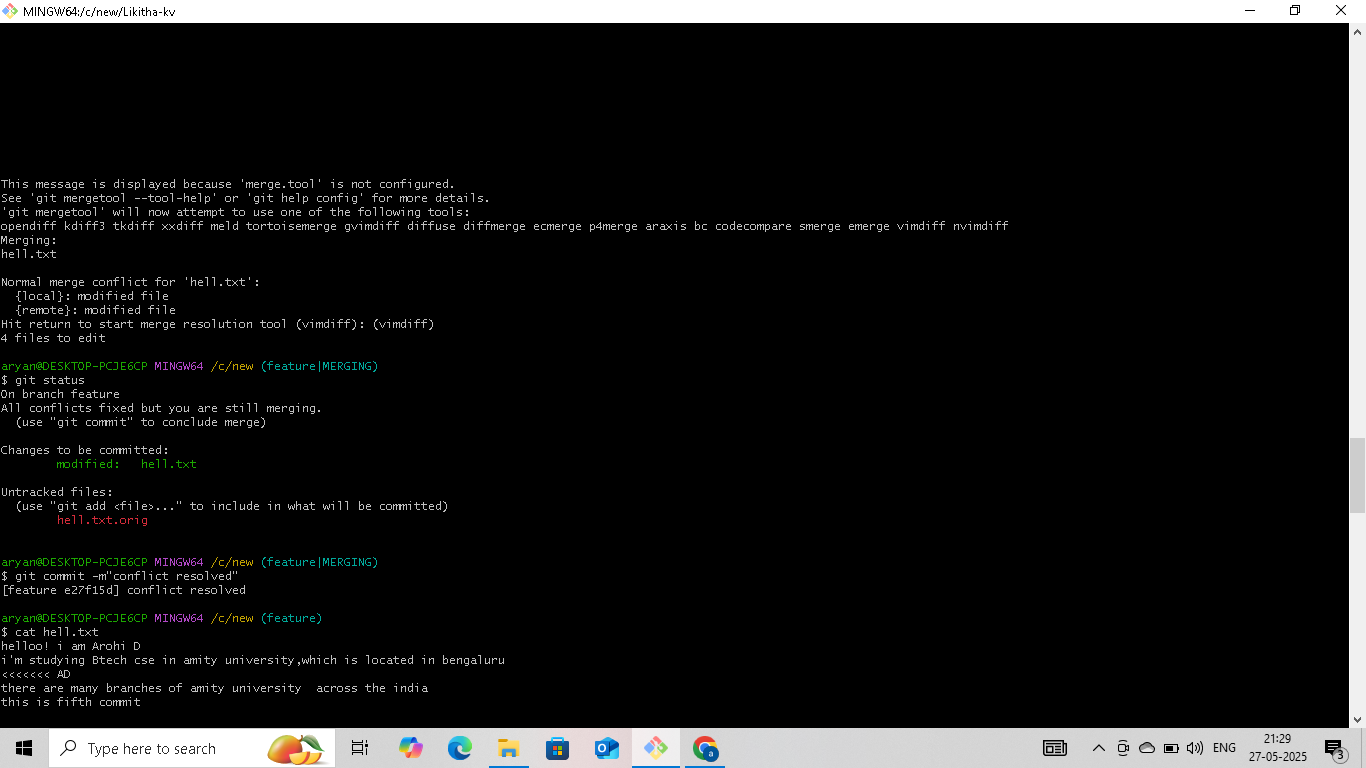
3.Create and Modify Files

* + - Create a file using vi command:

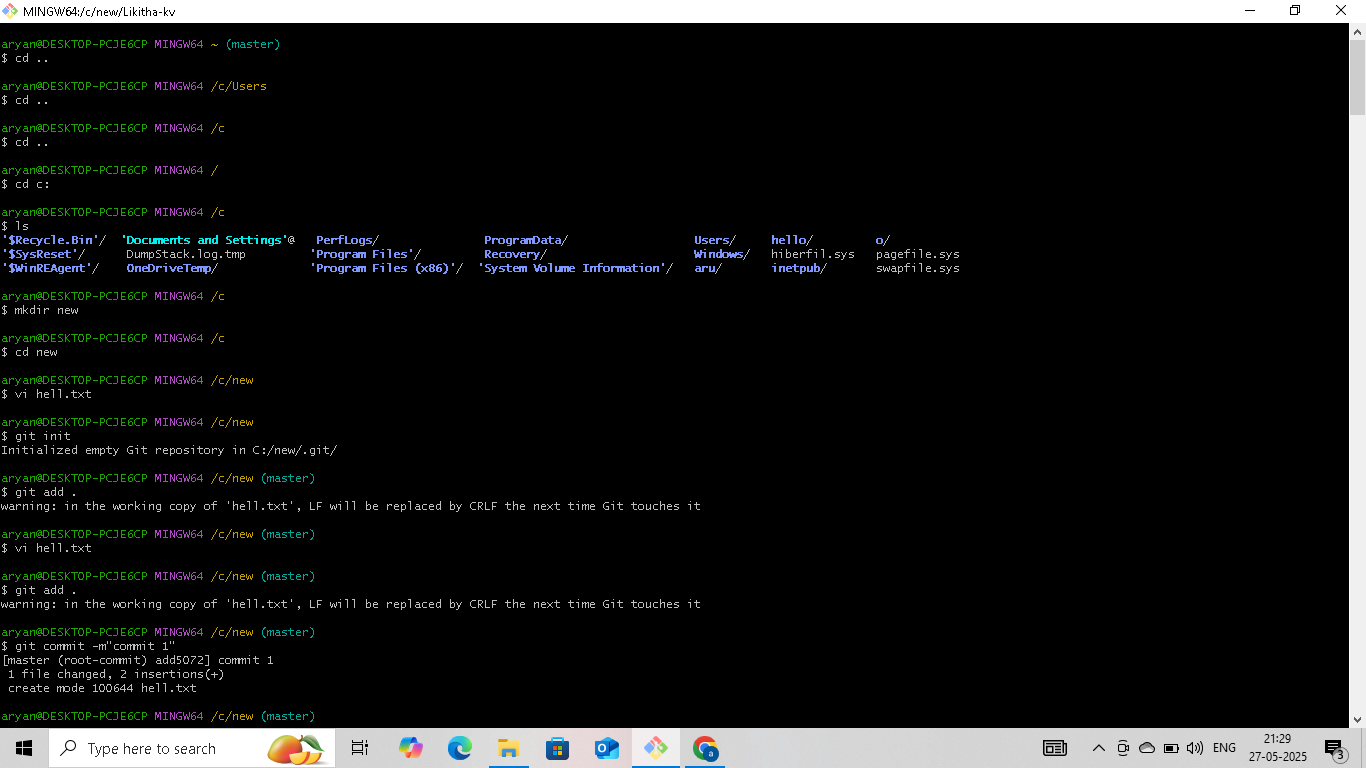


4.Check Repository Status

* + - View changes



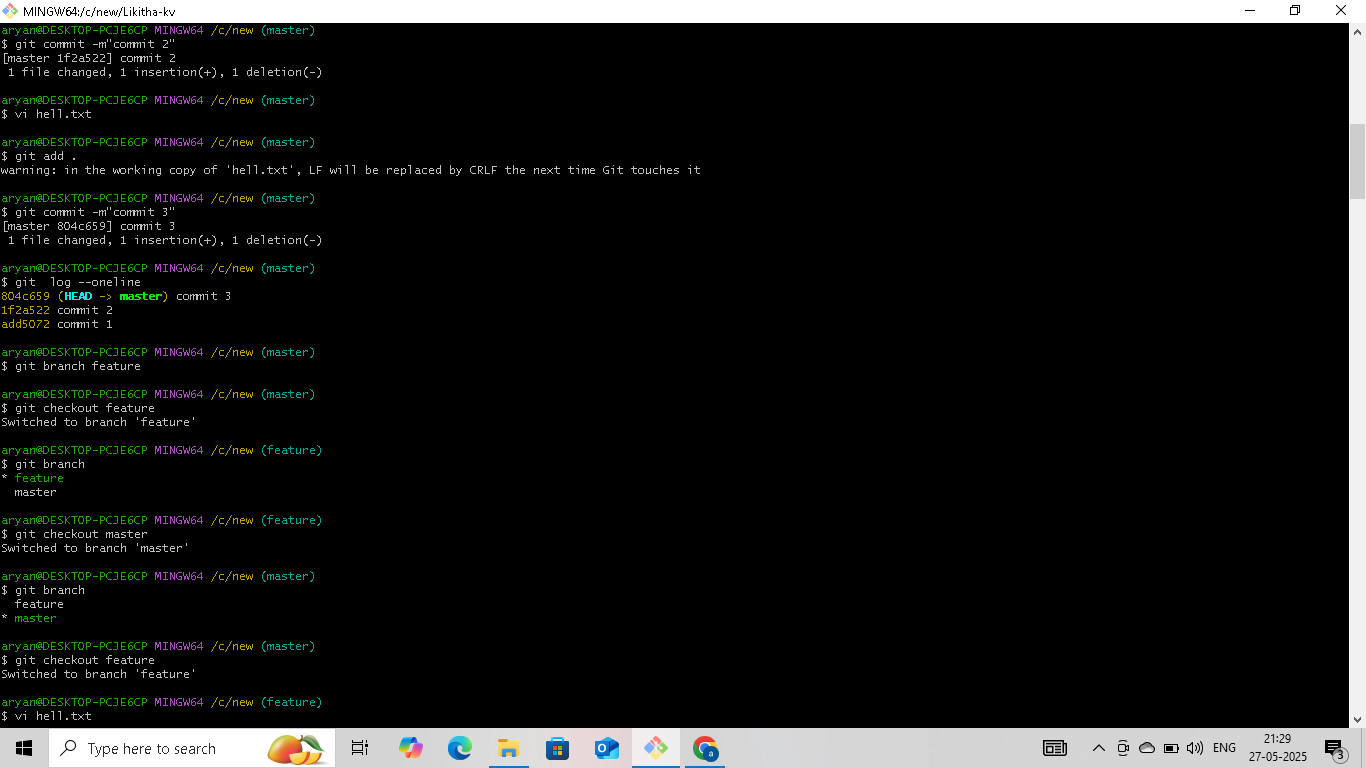
* 1. Stage and Commit Changes
     + Add files to the staging area
     + Commit changes:

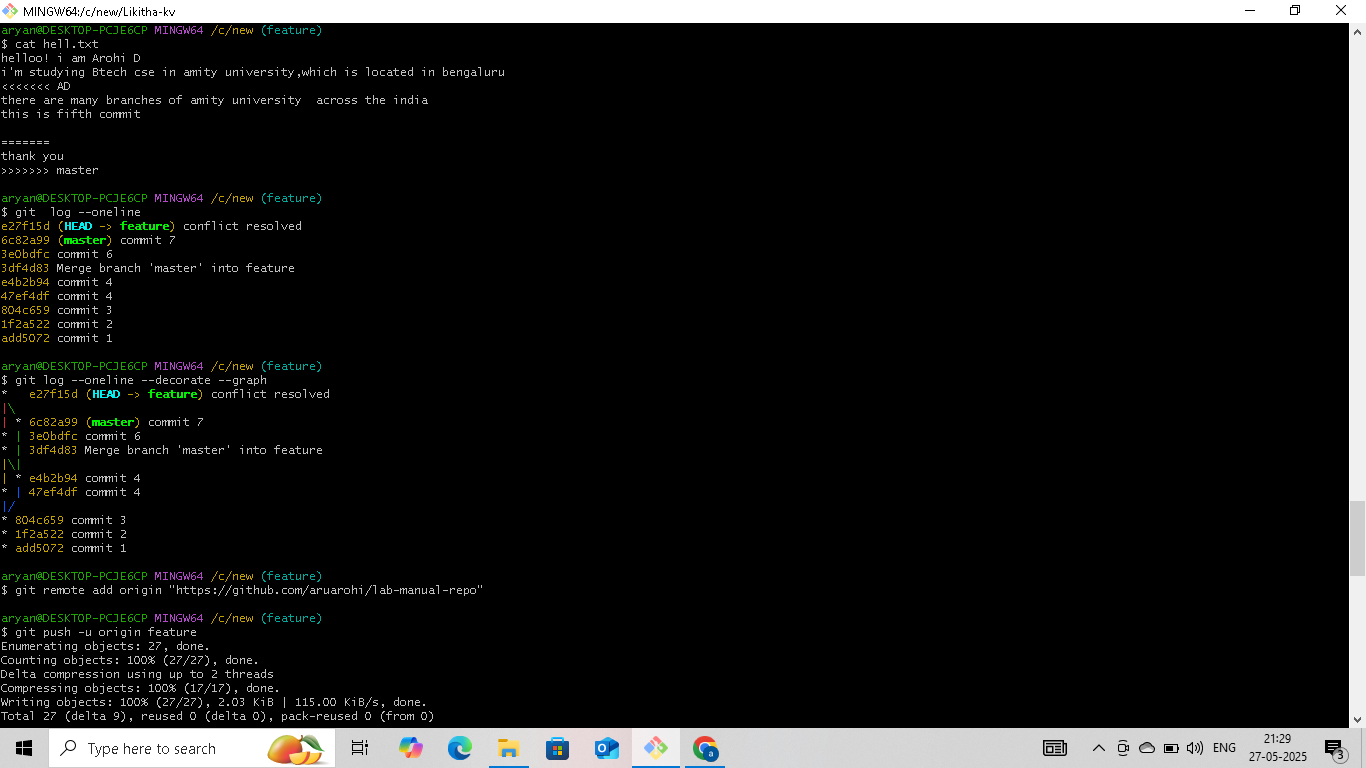


1. Create and Manage Branches

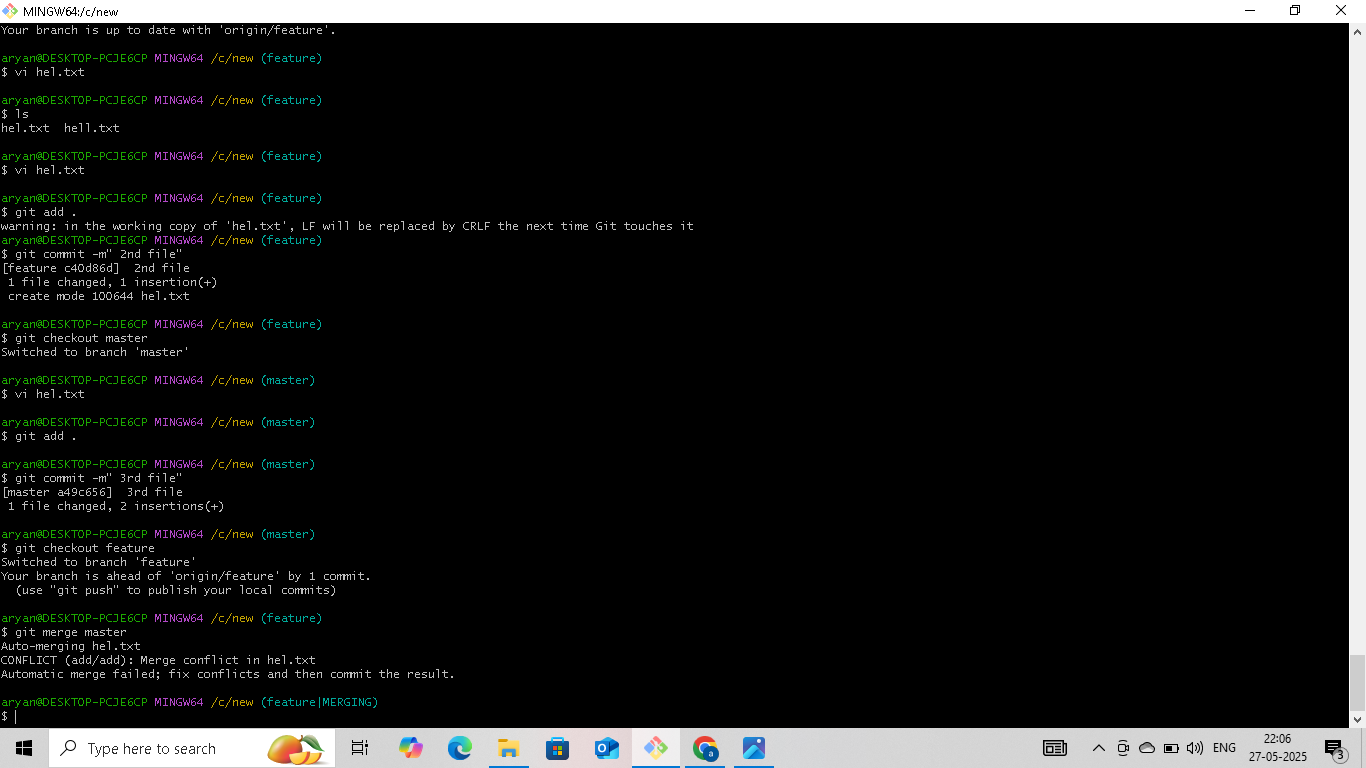
• Create a new branch:

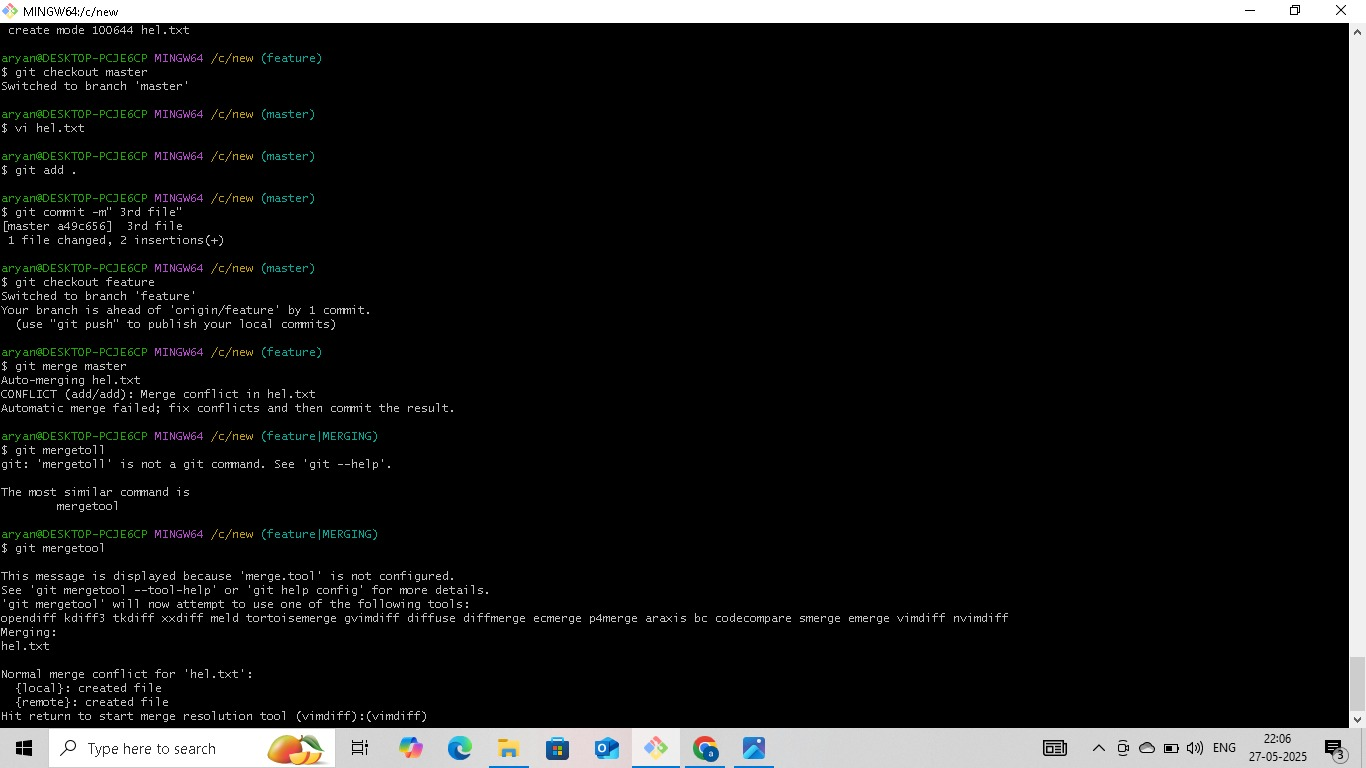
• Switch to the branch:



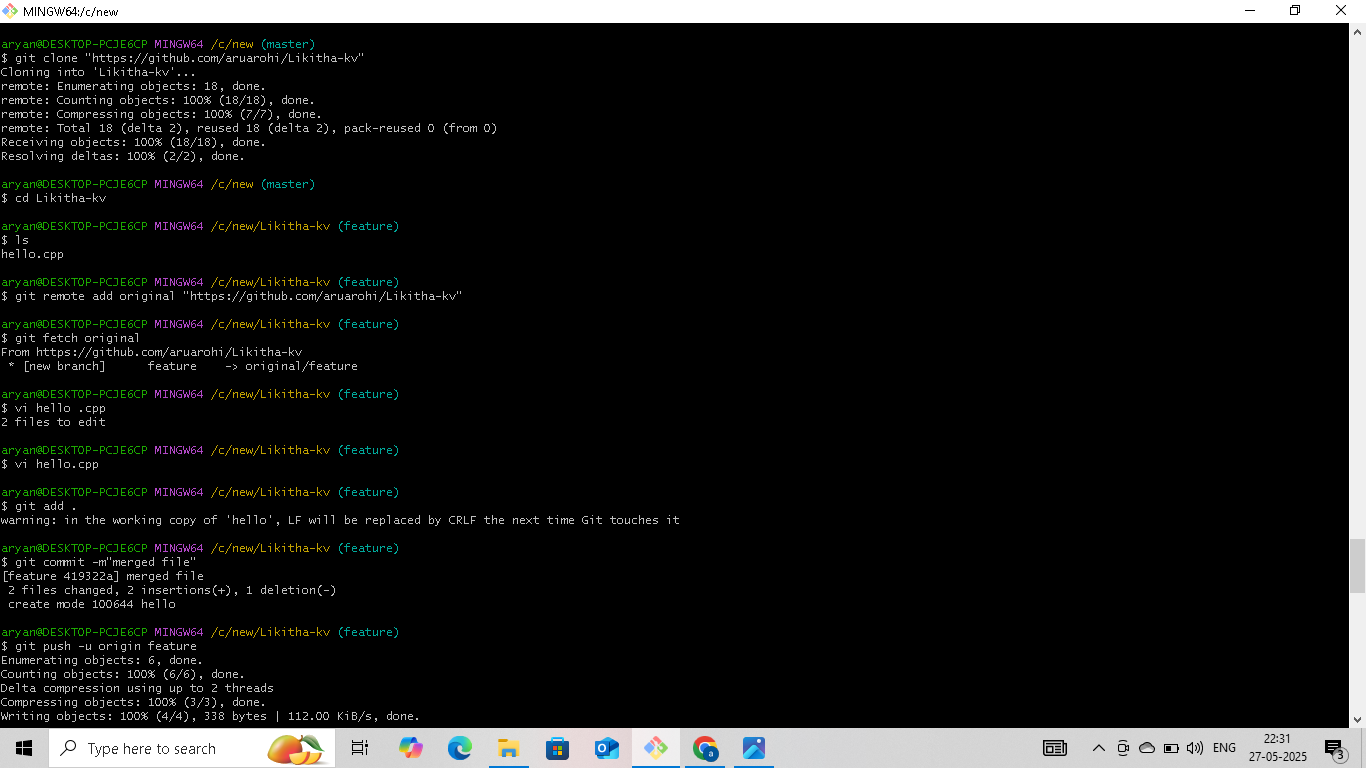


* + Merge the branch into the mainbranch:





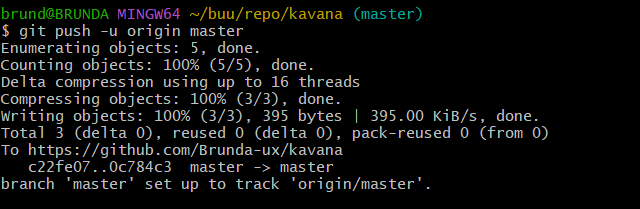
Git Clone:



1. Pull Changes from Remote Repository

• Update local repository:

8. Pull Changes from Remote Repository



10.Open and Close Pull Requests

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

