Source Code Management

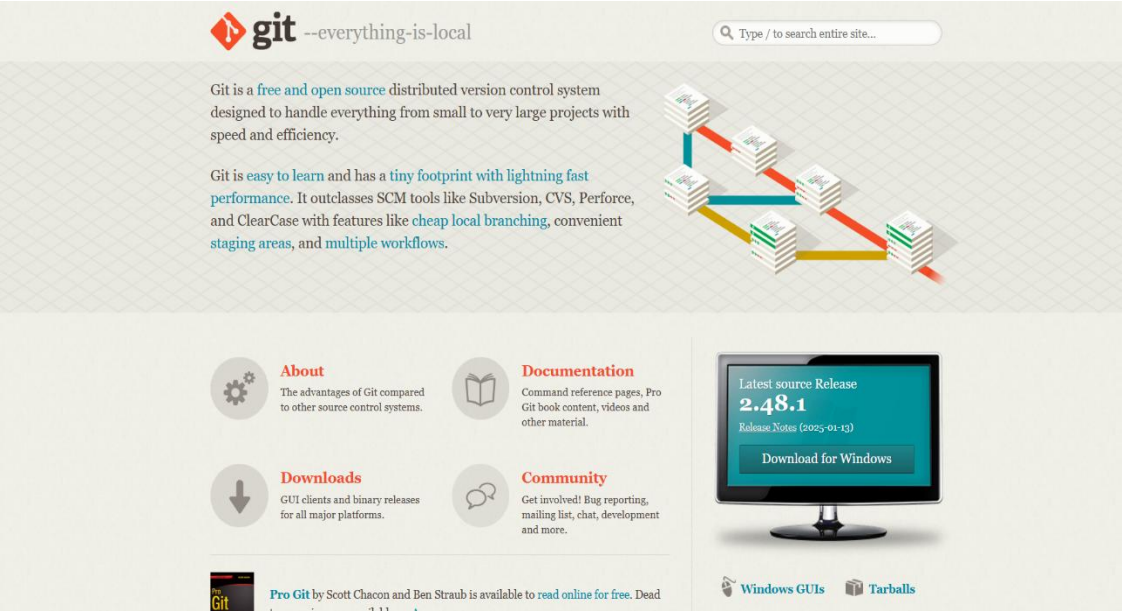
LAB REPORT – 1

Overview of Git:

Git is a distributed version control system that tracks versions of files. It is often used to control source code by programmers who are developing software collaboratively. Design goals of Git include speed, data integrity, and support for distributed, non-linear workflows — thousands of parallel branches running on different computers.

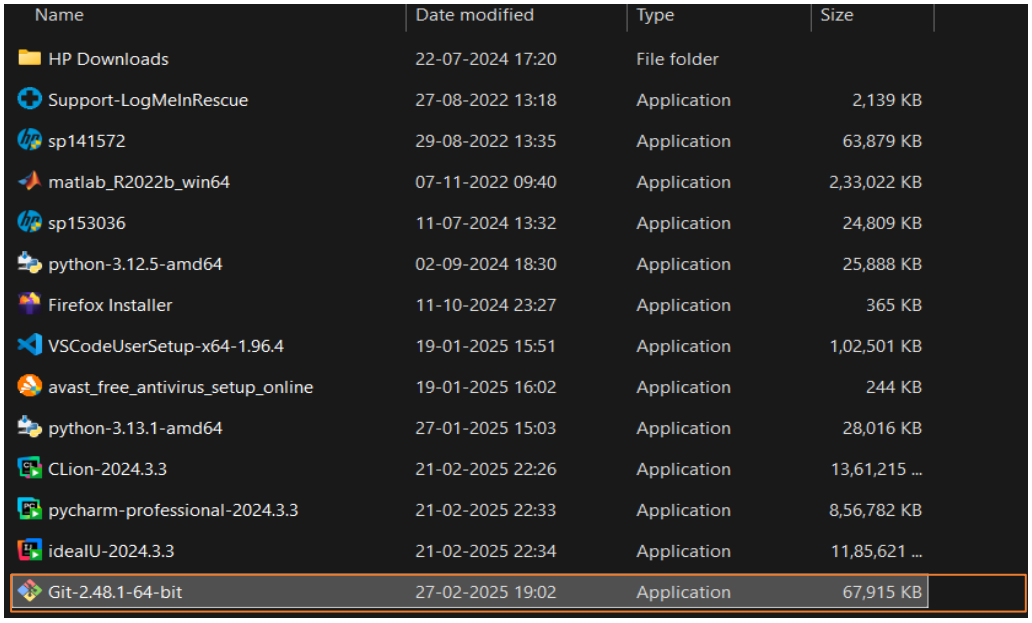
Step 1: Downloading Git

1. Open your web browser and navigate to the official Git website: https://git-scm.com.
2. On the homepage, you will see a "**Download**" button that automatically detects your OS. Click on the "Download" button to download the appropriate installer for your operating system (Windows, macOS, or Linux).
3. Alternatively, you can manually select your OS from the website to download a specific version.



Step 2: Running the Git Installer

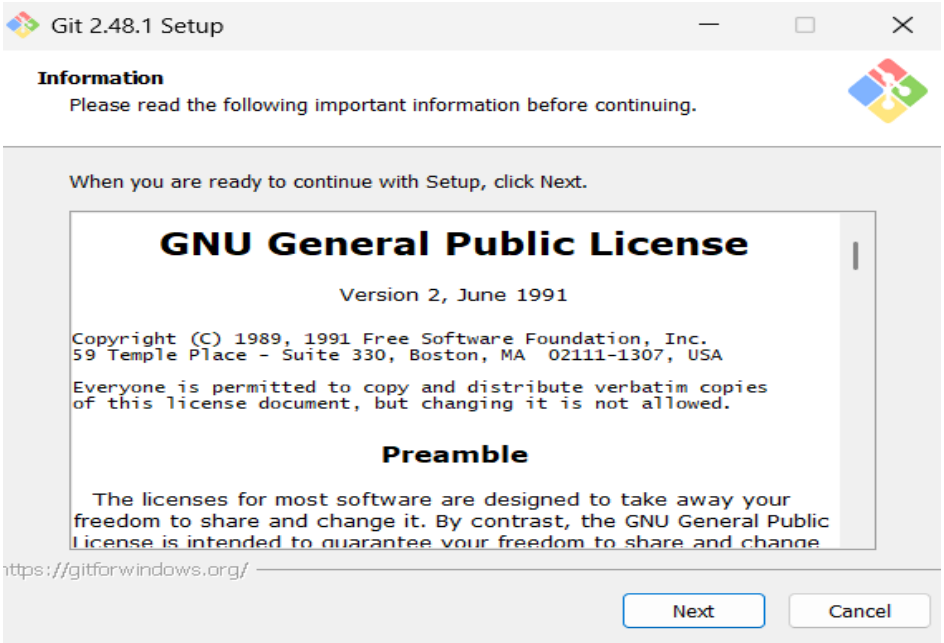
Locate the downloaded **Git.exe** file and double-click to run it.



Name	Date modified	Type	Size
HP Downloads	22-07-2024 17:20	File folder	
Support-LogMeInRescue	27-08-2022 13:18	Application	2,139 KB
sp141572	29-08-2022 13:35	Application	63,879 KB
matlab_R2022b_win64	07-11-2022 09:40	Application	2,33,022 KB
sp153036	11-07-2024 13:32	Application	24,809 KB
python-3.12.5-amd64	02-09-2024 18:30	Application	25,888 KB
Firefox Installer	11-10-2024 23:27	Application	365 KB
VSCodeUserSetup-x64-1.96.4	19-01-2025 15:51	Application	1,02,501 KB
avast_free_antivirus_setup_online	19-01-2025 16:02	Application	244 KB
python-3.13.1-amd64	27-01-2025 15:03	Application	28,016 KB
CLion-2024.3.3	21-02-2025 22:26	Application	13,61,215 ...
pycharm-professional-2024.3.3	21-02-2025 22:33	Application	8,56,782 KB
idealU-2024.3.3	21-02-2025 22:34	Application	11,85,621 ...
Git-2.48.1-64-bit	27-02-2025 19:02	Application	67,915 KB

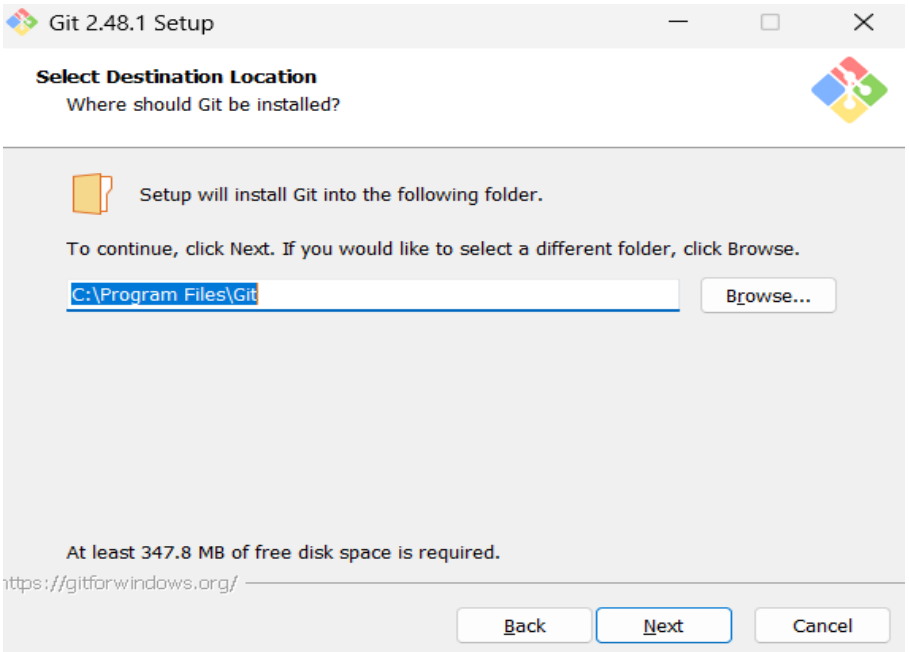
Step 3: License (Terms and Conditions)

Read the **GNU** General Public License's terms and conditions and click on **Next**.



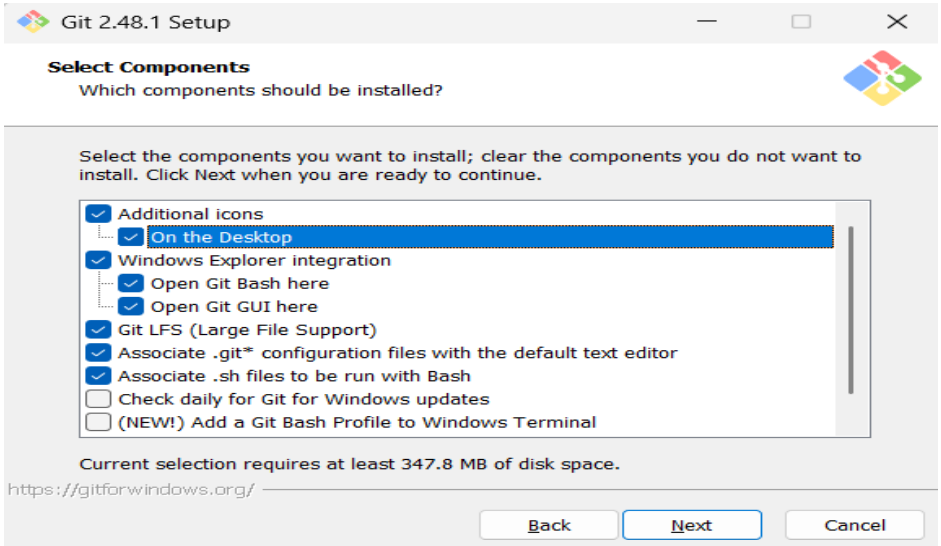
Step 4: Choose Installation Location

Choose the installation location (default is **C:\Program Files\Git**) and click **Next**.



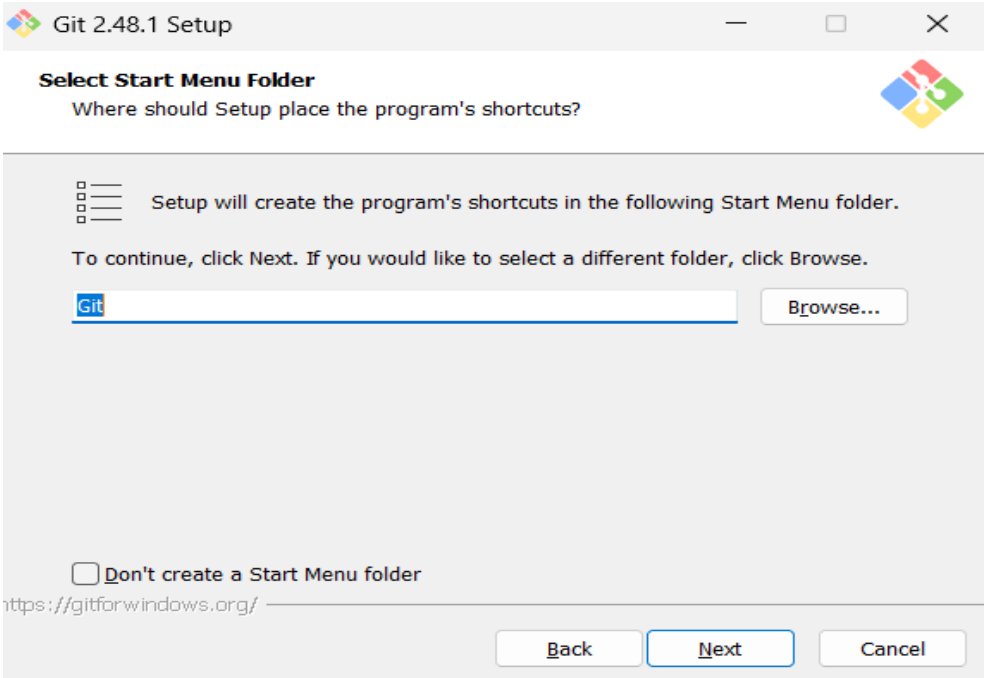
Step 5: Select the Components

Select the components you want (default options are fine) and click **Next**.



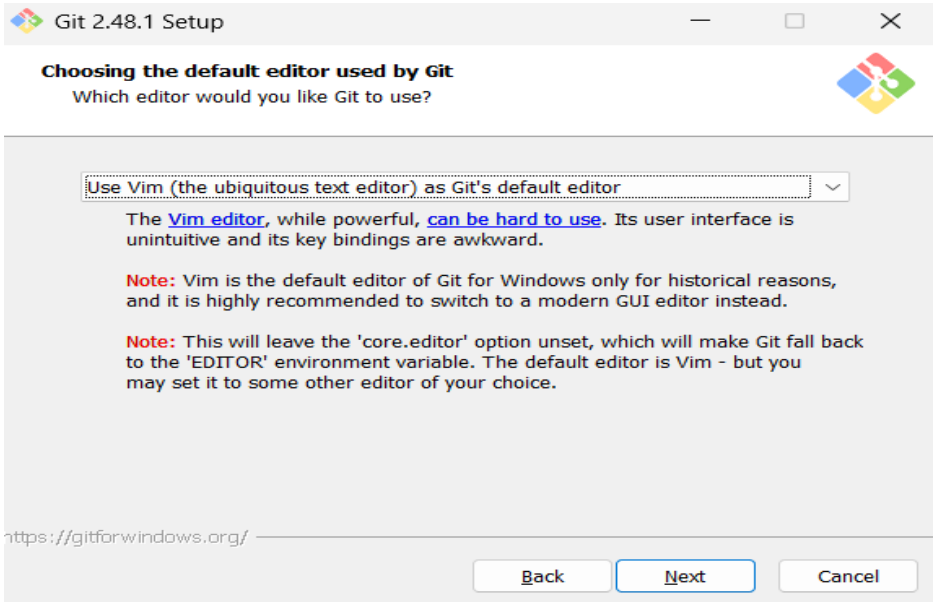
Step 6: Select Start Menu Folder

Choose the Start Menu folder where Git shortcuts will be placed. By default, the folder is named **"Git"**. Keep the default name and click **Next** to Proceed.



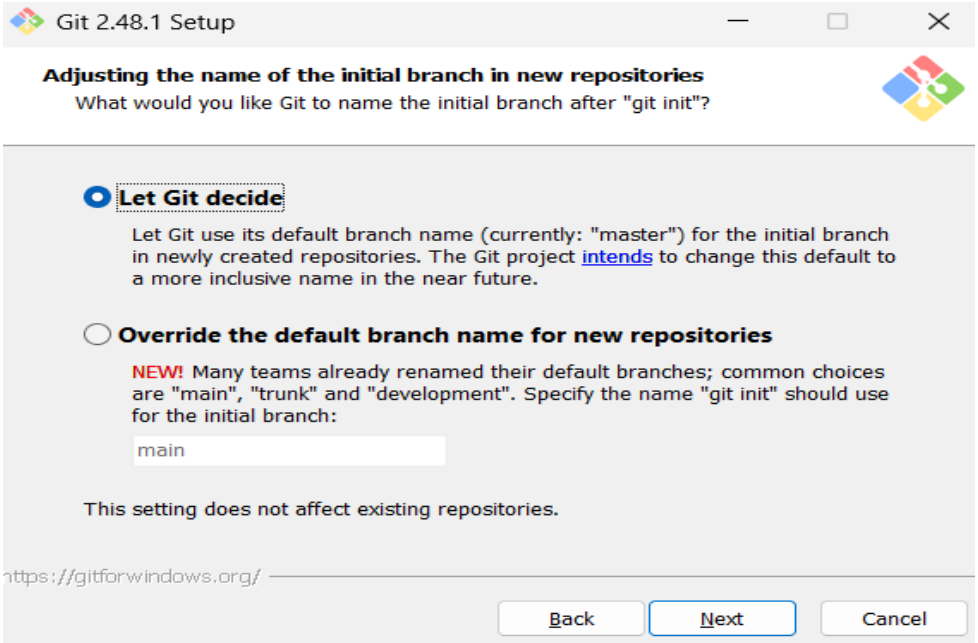
Step 7: Choose the Text Editor

Choose a default text editor (select **Vim**) and Click **Next**.



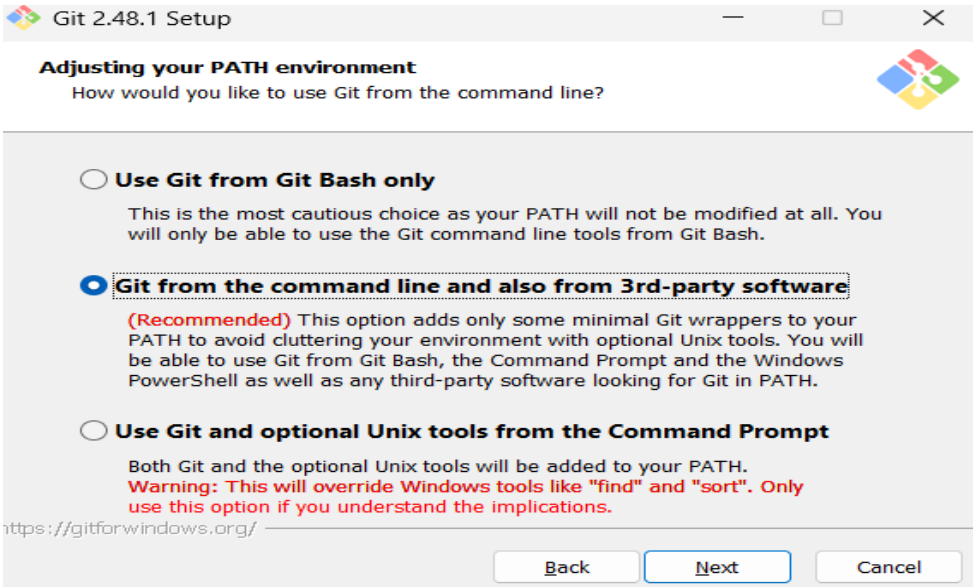
Step 8: Adjusting Initial Branch Name

Choose the default name for the first branch when initializing a new Git repository. Go with **'Let Git Decide'** option setting the branch as **Master** branch and proceed with **Next**.



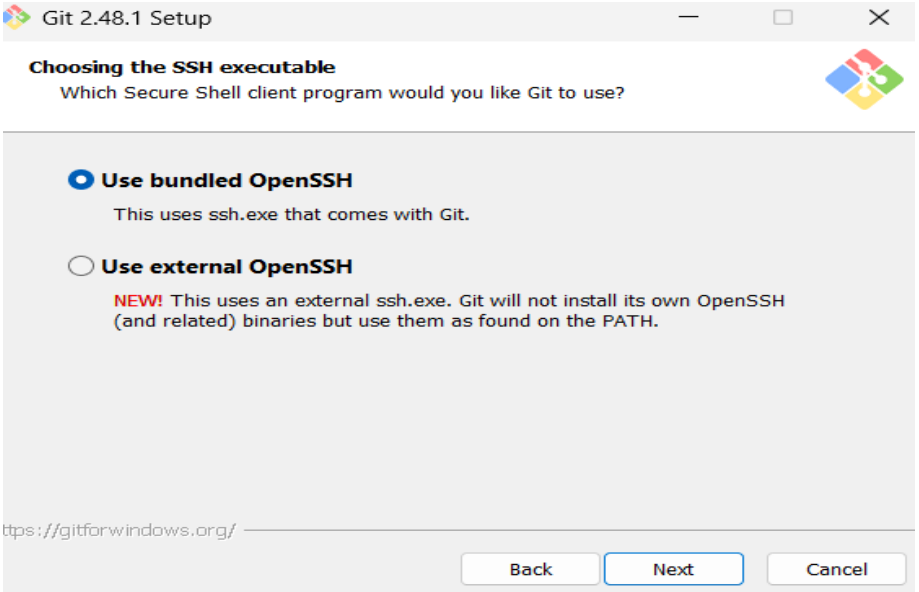
Step 9: Adjusting PATH Environment

Select **Git from the command line and also from third-party software** (recommended). Click **Next**.



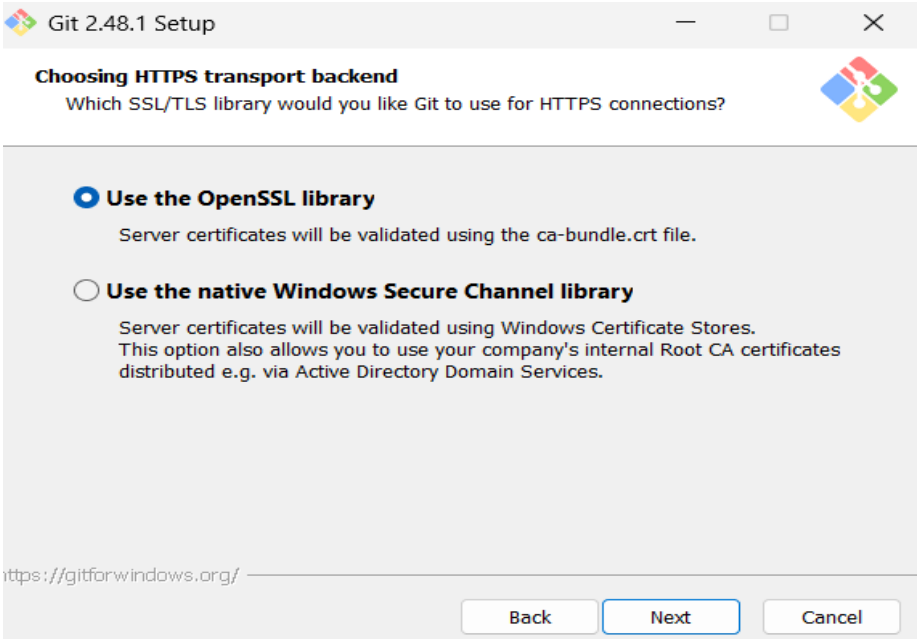
Step 10: Choosing the SSH Executable

Select "**Use bundled OpenSSH**" for better compatibility and Click on **Next**.



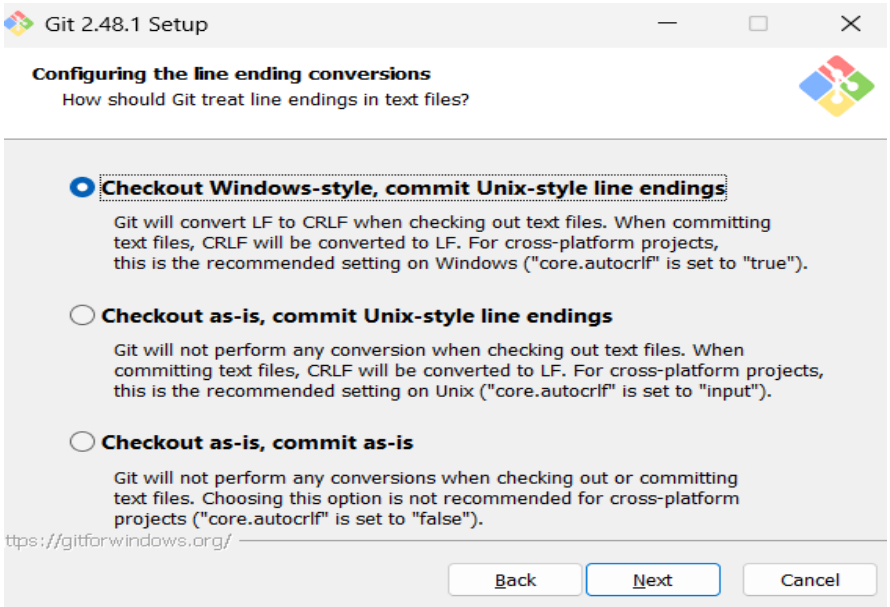
Step 11: Choosing the HTTP Transport Background

Choose Use the **OpenSSL library** (default) and Click **Next**.



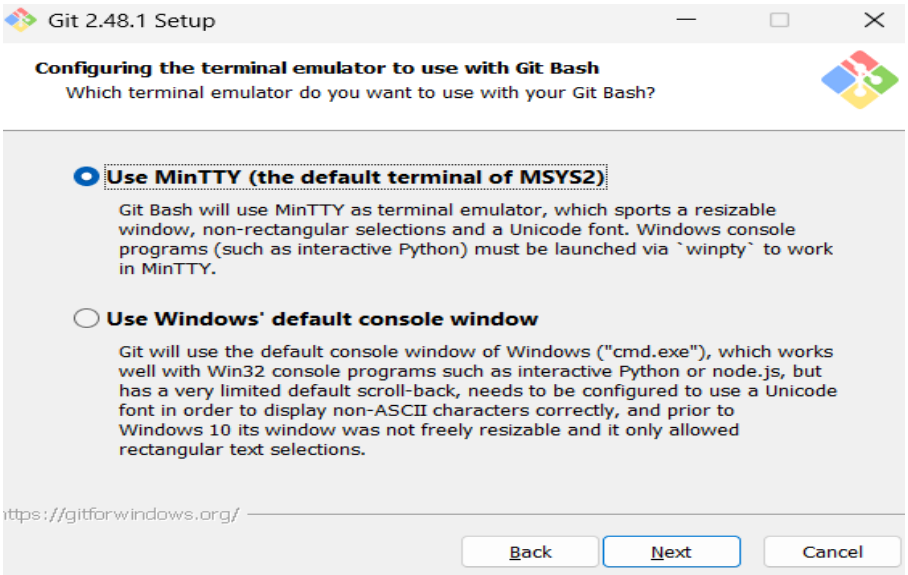
Step 12: Configuring Line Ending Configs

Select **Checkout Windows-style, commit Unix-style line endings** (recommended) and Click **Next**.



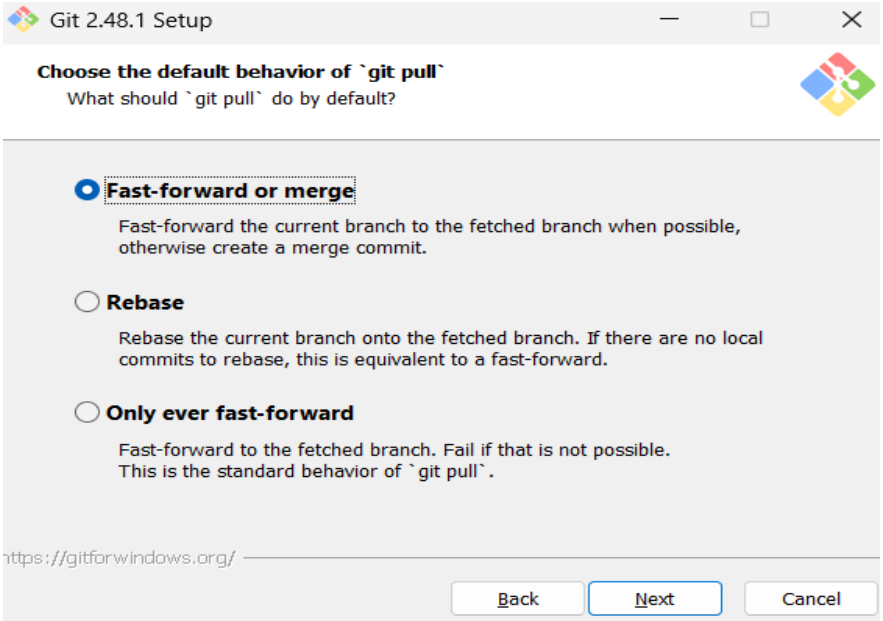
Step 13: Configuring the Terminal Emulator

Select **Use MinTTY (default terminal for MSYS2**) and Click **Next**.



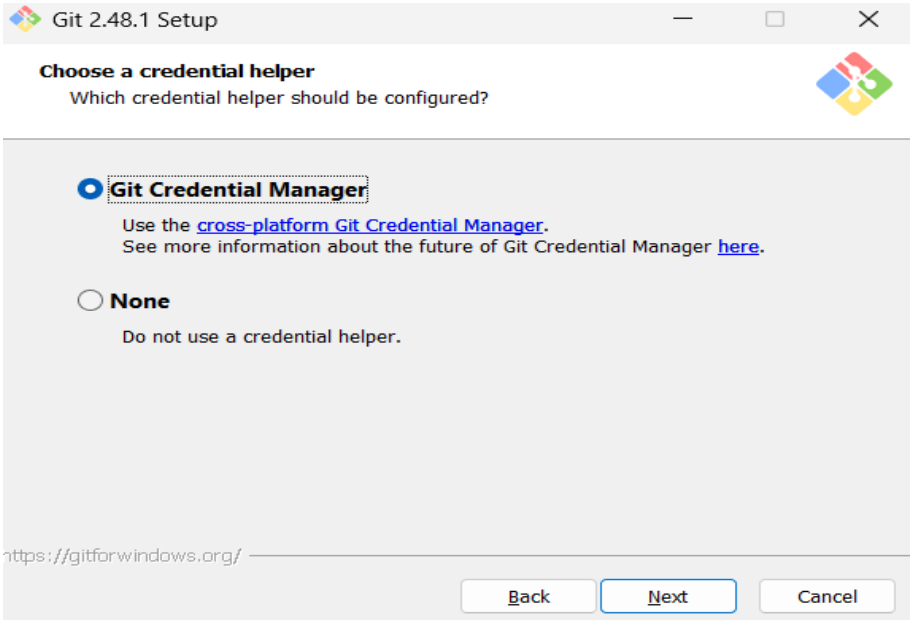
Step 14: Choosing the Default Behaviour

Select **Fast-forward or Merge** (recommended) option and click **Next**.



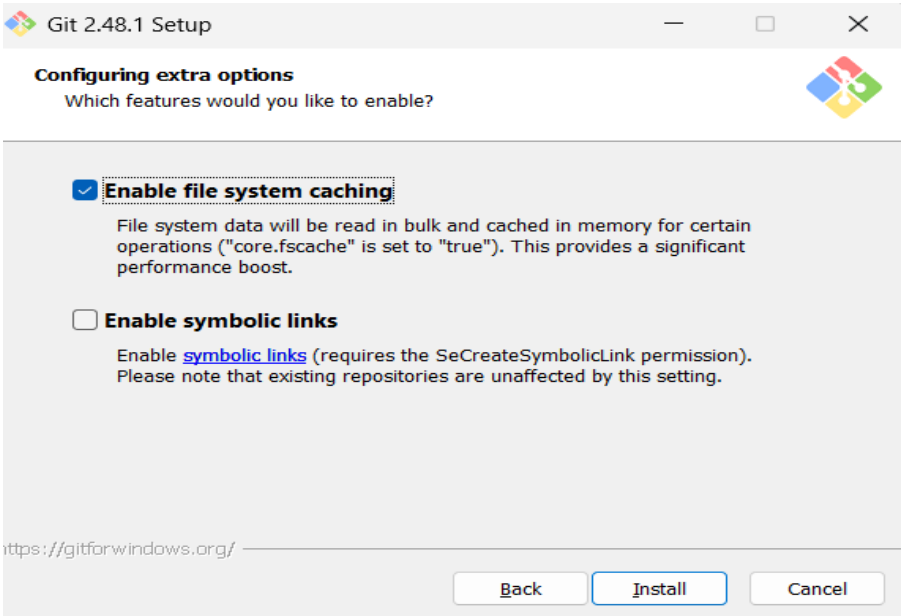
Step 15: Choosing a Credential Helper

Select **Git Credential Manager** (recommended) and Click **Next**.



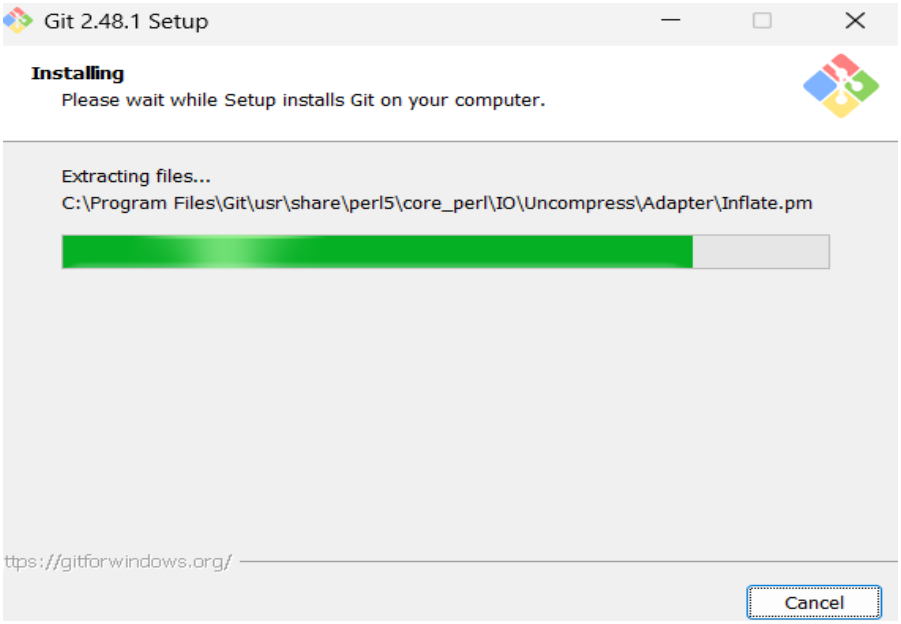
Step 16: Configuring Extra Options

Select **Enable file system caching** (recommended) and Click on **Install**.



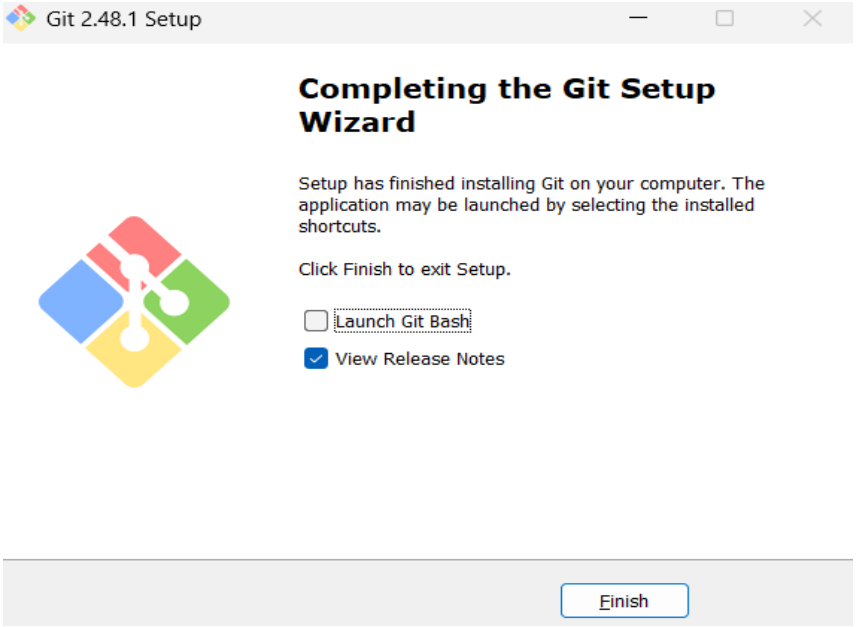
Step 17: Installation Overview

A progress bar (**green bar**) will appear, indicating that Git is being installed. Wait for the installation to complete. This may take a few minutes.



Step 18: Completing the Git Set - Up Wizard

Once the installation is complete, **"Completing the Git Setup Wizard"** screen appears. Check the '**Launch Git bash'** option and Click on **Finish**.



Source Code Management

LAB REPORT – 2

Step 1: Open Git Bash

Open **Git Bash** from the Start menu or by searching for it.

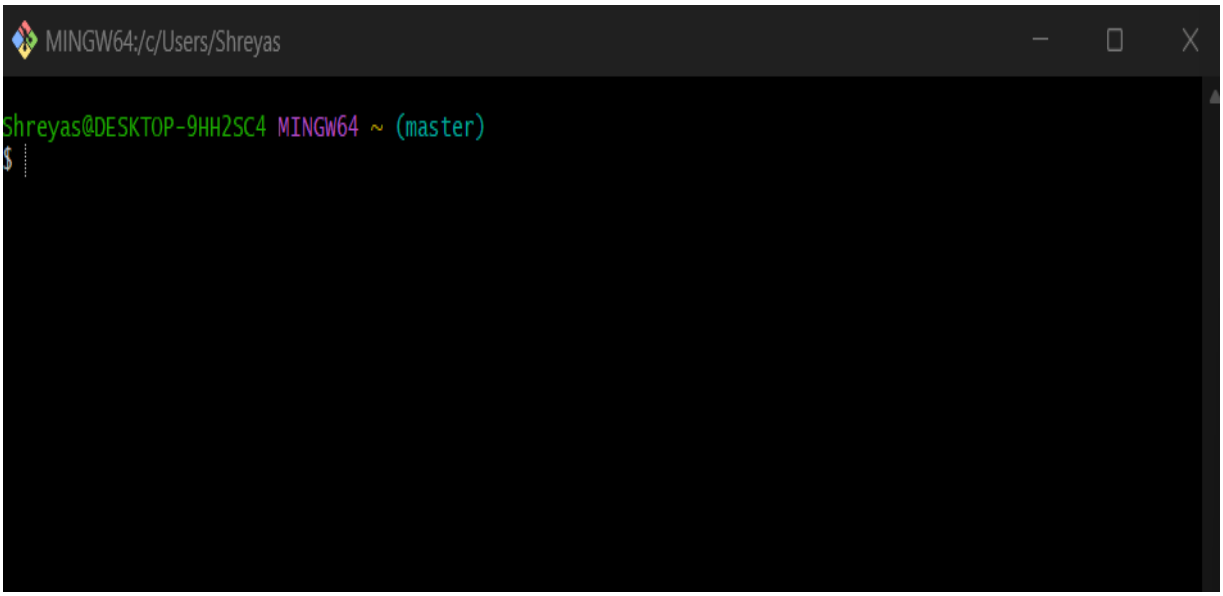


Figure – 1

Step 2: Check Git Version

To verify that Git is installed correctly, run: **git –version**

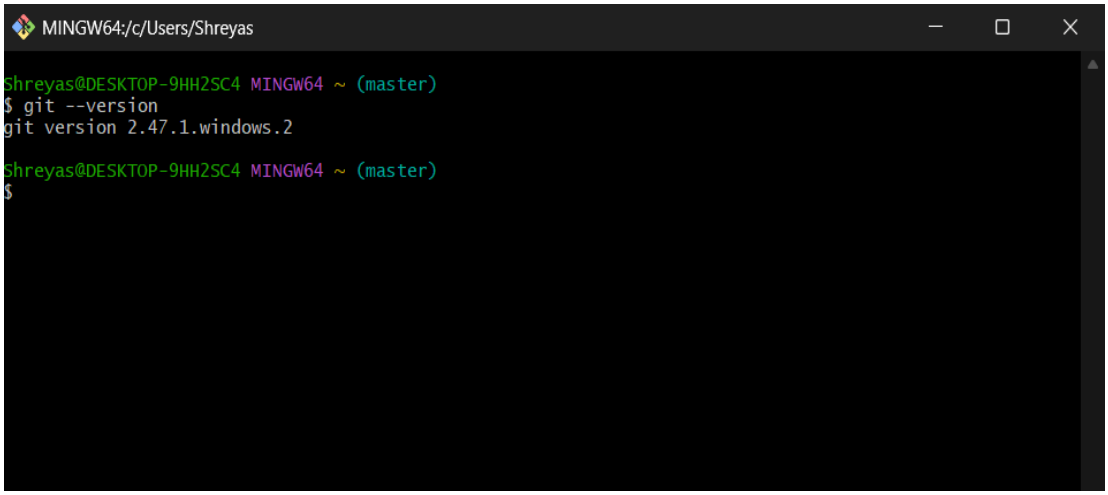


Figure – 2

Step 3: Configure Git

Set up your Git username and email (required for commits):

- **git config --global user.name "Your Name"**
- **git config --global user.email “your-email@example.com”**

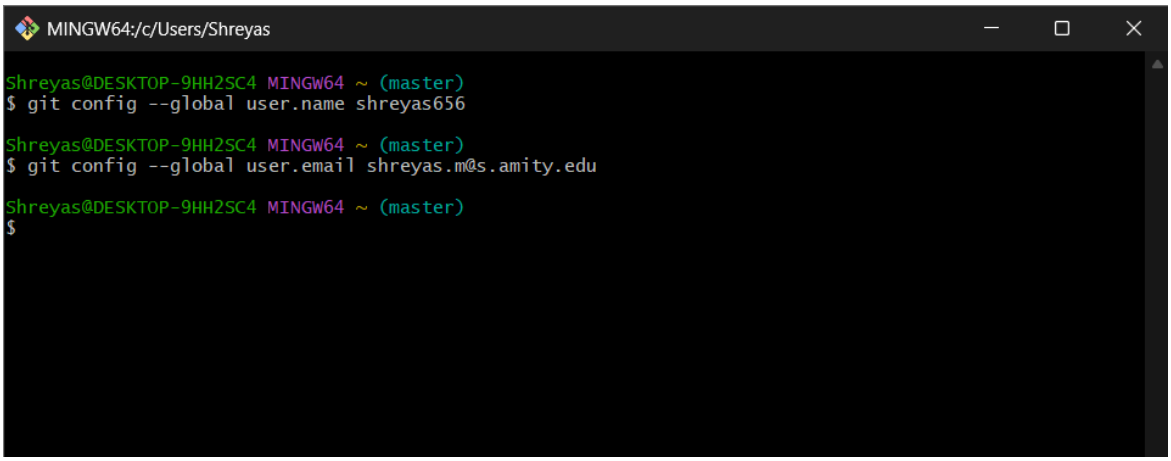


Figure – 3

Step 4: Verify Git Configurations

To check if the configurations were set correctly, run:

- **git config --list**

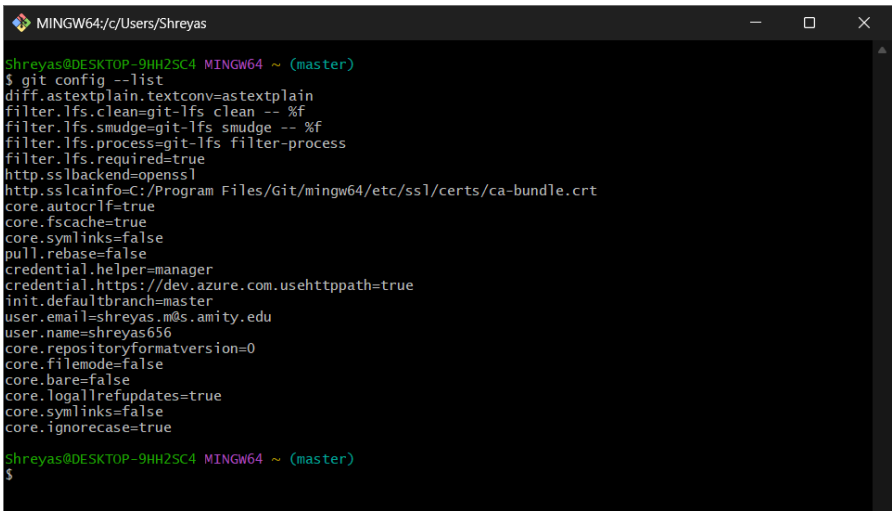


Figure – 4

Step 5: Change Directory

Change directory (**cd**) to your preferred location using the **'cd'** command.

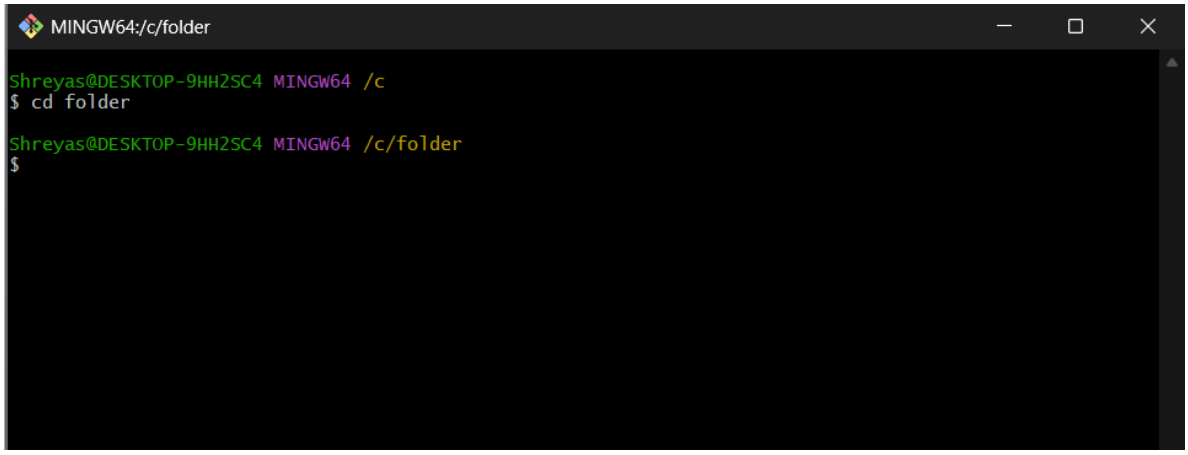


Figure – 5

Step 6: Print the Current Directory

To print the full path of your current Directory use the '**pwd'** command.

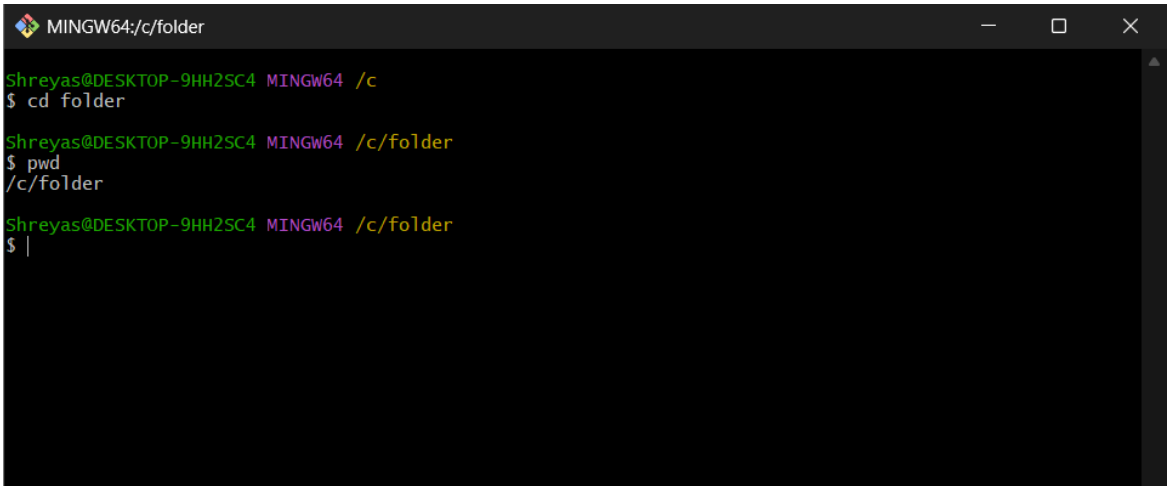


Figure – 6

Step 7: Create a New Folder

To Create a new folder in the Directory, use the command: **mkdir** folder-name.

Figure – 7

Step 8: Listing the Files and Folders

To Display the list of all files and folders in the current directory use the **'ls'** command.

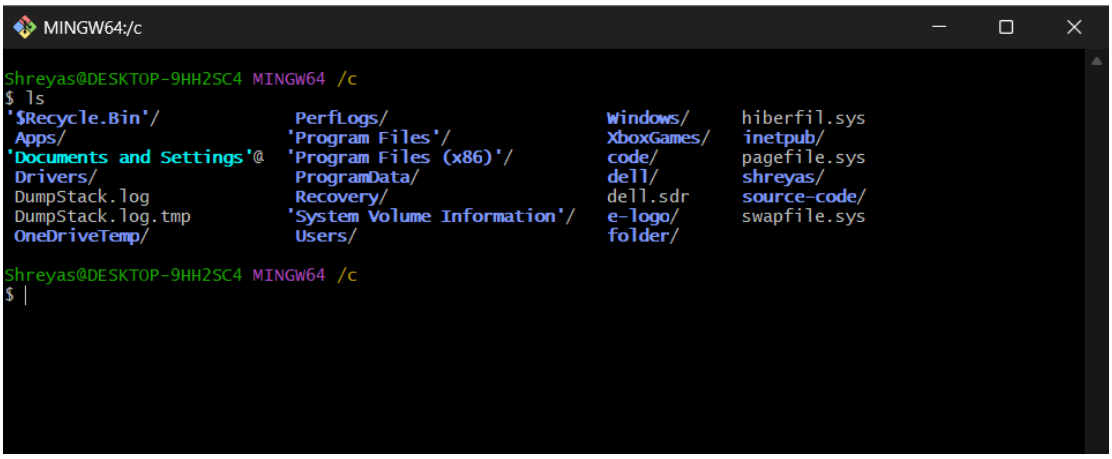


Figure – 8

Step 9: Creating a File Inside the Folder

To create a C++ File inside the **Git** Folder, move inside the folder using the '**cd**' command and then use **'vi'** command to create a file.

```
MINGW64:/c/folder

Shreyas@DESKTOP-9HH2SC4 MINGW64 /c
$ ls
'$Recycle.Bin'/            PerfLogs/                  Windows/     hiberfil.sys
 Apps/                    'Program Files'/            XboxGames/   inetpub/
'Documents and Settings'@  'Program Files (x86)'/     code/        pagefile.sys
 Drivers/                  ProgramData/               dell/        shreyas/
 DumpStack.log             Recovery/                  dell.sdr     source-code/
 DumpStack.log.tmp        'System Volume Information'/ e-logo/    swapfile.sys
 OneDriveTemp/             Users/                     folder/

Shreyas@DESKTOP-9HH2SC4 MINGW64 /c
$ cd folder

Shreyas@DESKTOP-9HH2SC4 MINGW64 /c/folder
$ |
```

Figure – 9

Step 10: Inside the VI Editor

Once typed Git opens the **'vi'** editor to create or edit a file named **Hello.cpp.** Press **i** to enter **INSERT** mode. Now start typing your code in the **vi** Editor.

```
MINGW64:/d/Git
#include <iostream>
using namespace std;

int main() {
    cout << "Hello, Git!" << endl;
    return 0;
}
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
Hello.cpp[+] [unix] (05:29 01/01/1970)                    7,2 All
-- INSERT --
```

Figure – 10

Step 11: Exiting the VI Editor

Once done with the code Press **ESC** to exit **INSERT** mode and type **:wq** and press **Enter** to save and exit.

```
MINGW64:/d/Git
$ cd /d

greyh@Nisanth-Notebook-25 MINGW64 /d
$ pwd
/d

greyh@Nisanth-Notebook-25 MINGW64 /d
$ mkdir Git

greyh@Nisanth-Notebook-25 MINGW64 /d
$ ls
'$RECYCLE.BIN'/          MapData/                    WpSystem/
'Data Vault'/           'Program Files'/            'avast! sandbox'/
 DeliveryOptimization/  'System Volume Information'/  hp/
 Git/                    WindowsApps/

greyh@Nisanth-Notebook-25 MINGW64 /d
$ cd /d/Git

greyh@Nisanth-Notebook-25 MINGW64 /d/Git
$ vi Hello.cpp

greyh@Nisanth-Notebook-25 MINGW64 /d/Git
$
```

Figure – 11

Step 12: Display File Contents

To Display the contents of the CPP File use the **cat** Command as: **cat** filename.extension.

```
MINGW64:/d/Git
$ ls
'$RECYCLE.BIN'/          MapData/                    WpSystem/
'Data Vault'/           'Program Files'/            'avast! sandbox'/
 DeliveryOptimization/  'System Volume Information'/  hp/
 Git/                    WindowsApps/

greyh@Nisanth-Notebook-25 MINGW64 /d
$ cd /d/Git

greyh@Nisanth-Notebook-25 MINGW64 /d/Git
$ vi Hello.cpp

greyh@Nisanth-Notebook-25 MINGW64 /d/Git
$ cat Hello.cpp
#include <iostream>
using namespace std;

int main() {
    cout << "Hello, Git!" << endl;
    return 0;
}

greyh@Nisanth-Notebook-25 MINGW64 /d/Git
$
```

Figure – 12

Step 13: Initialize Git in Directory

To turn the directory into a Git repository, run: **git init**

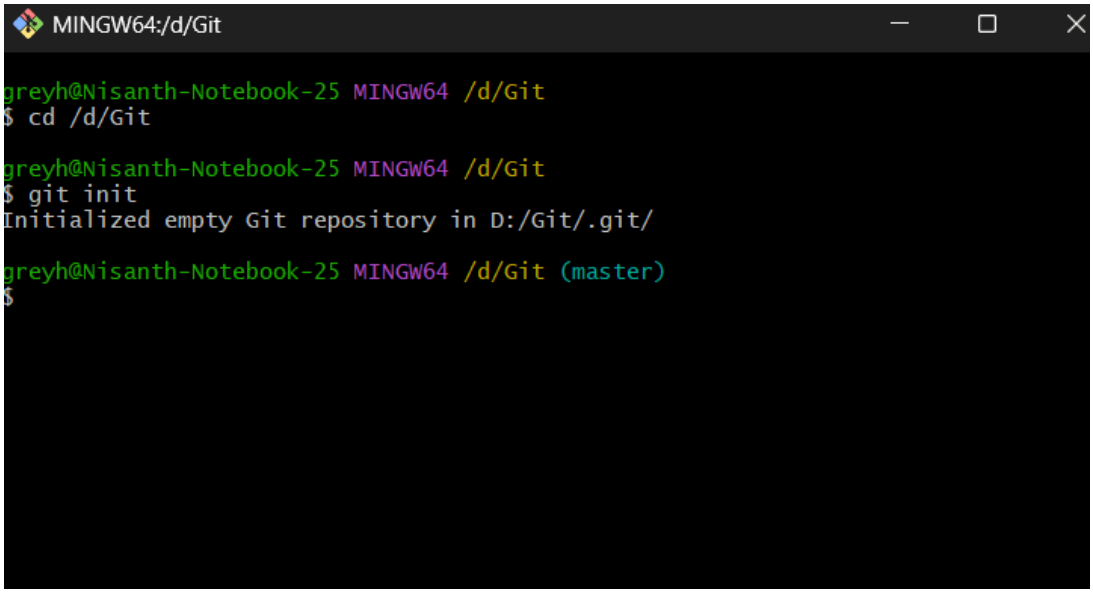


Figure – 13

Step 14: Check Git Status

The **git status** command is used to check for **untracked files**, along with other changes in the repository. You should see Hello.cpp as an **untracked file**.

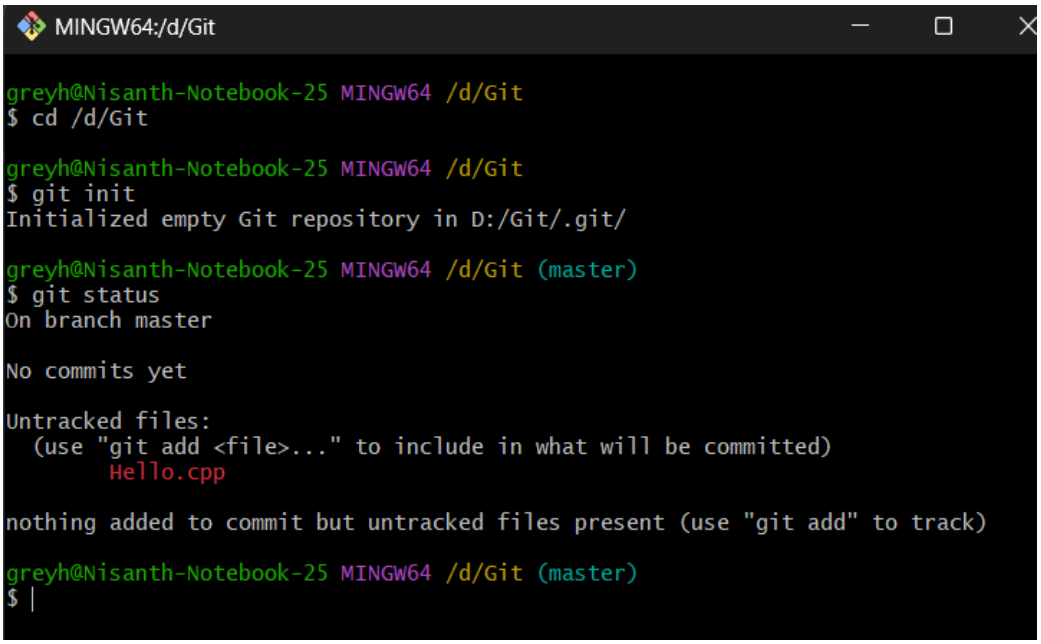


Figure – 14

Step 15: Add Files to Staging Area

To stage all newly created and modified files use the command: **git add .**

To confirm, check the status again using the command: **git status**

Now, all tracked files will appear as **staged**.

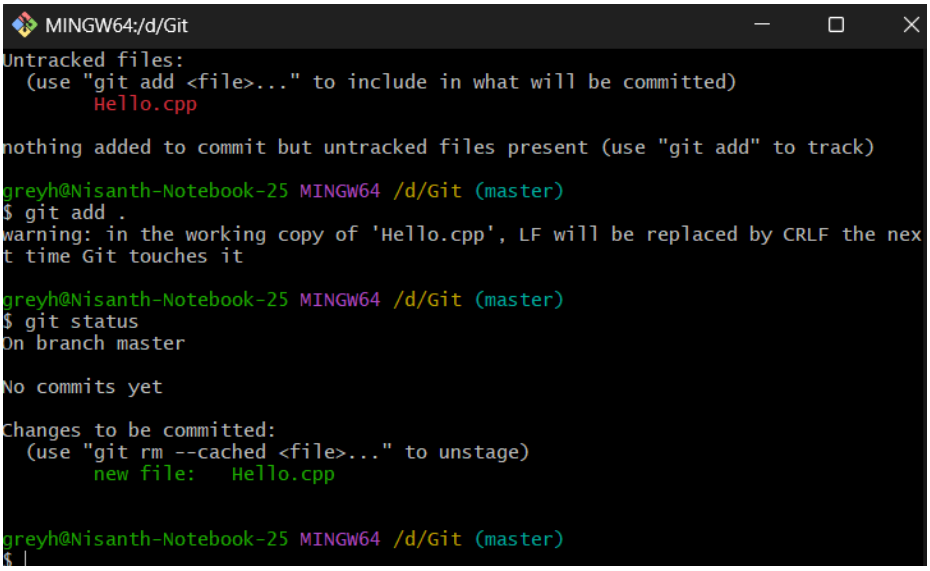


Figure – 15

Step 16: Commit the File

To save the changes in Git, commit the file with a message**: git commit -m "Initial commit: Added main.cpp"**

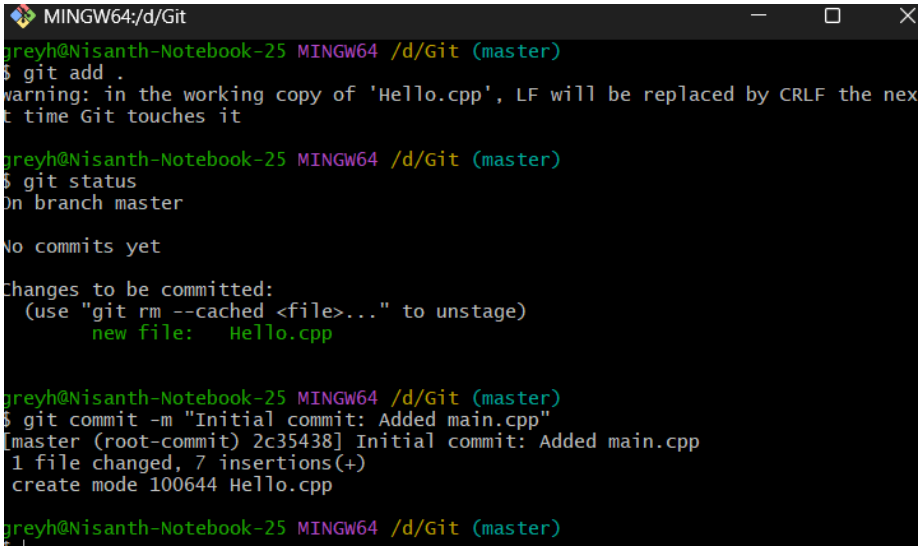


Figure – 16

Source Code Management

LAB REPORT – 3

Step 1: Check Git Commit History

- The **git log** command displays the commit history in detail.
- It shows the commit hash, author, date, and commit message.

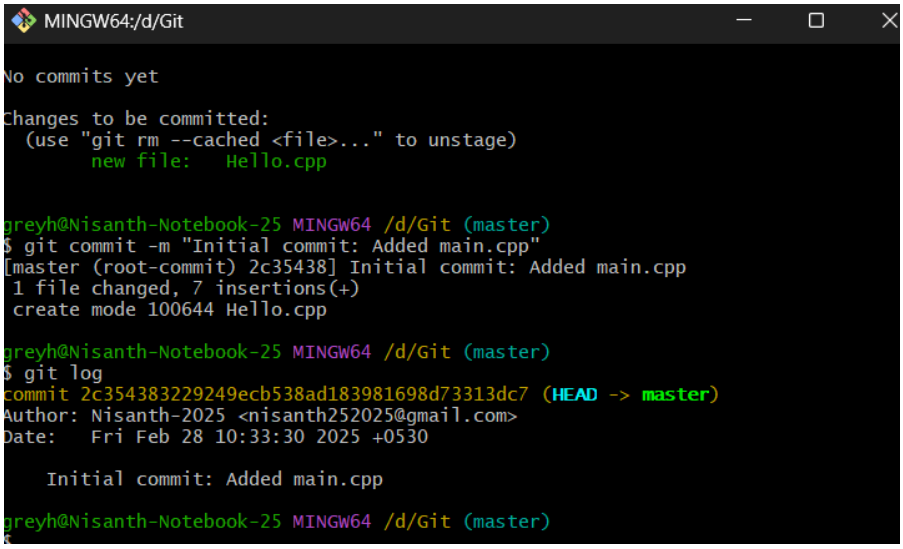


Figure – 1

Step 2: View Git Log in One Line Format

- The **git log --oneline** command displays a compact version of the commit history.
- It only shows the commit hash and the commit message.

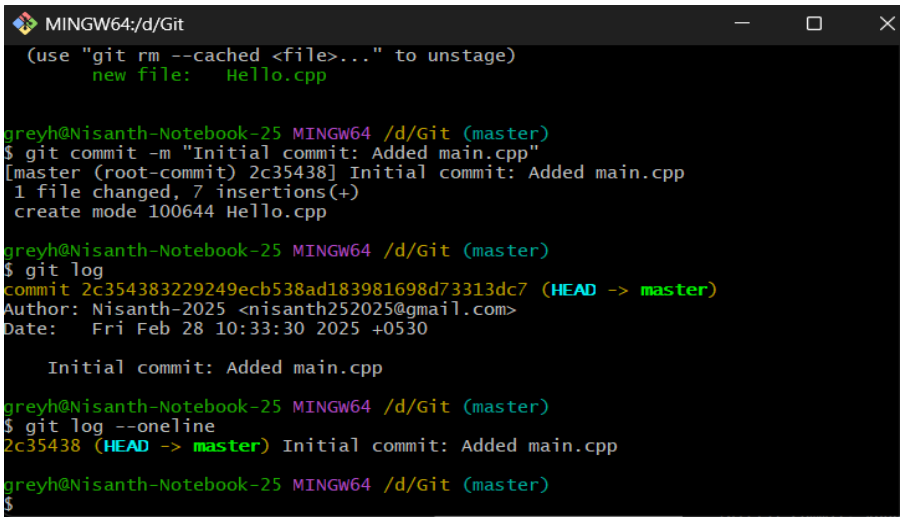


Figure – 2

Step 3: Modify the Hello.cpp File (First Change)

- Open the Hello.cpp file in a text editor using the **vi** command.
- Make a small change (e.g., add a new function or modify a print statement).
- Save the file and display it using the **cat** command.

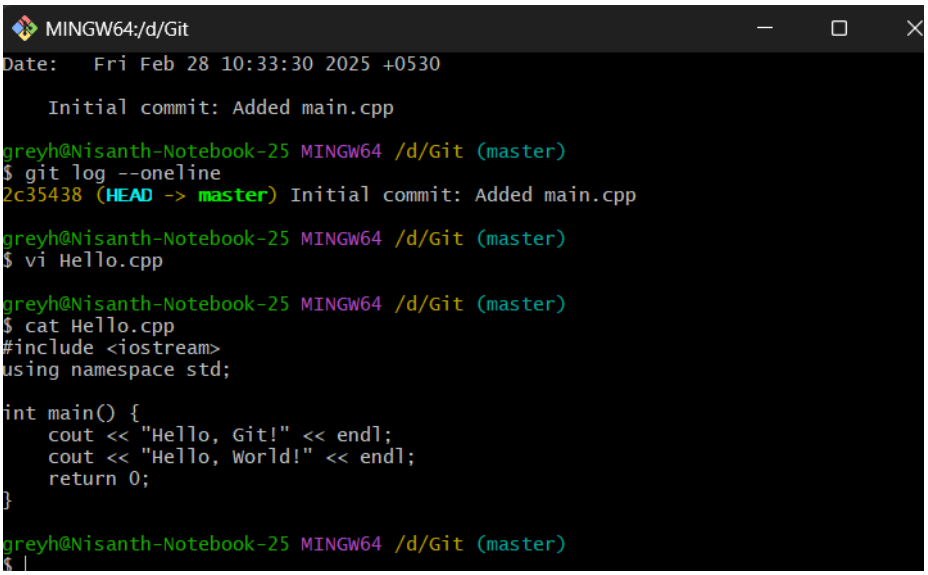


Figure – 3

Step 4: Stage and Commit the First Change

Use **git add .** command to stage the modified file for commit **and git commit -m** to create a commit with a message describing the change.

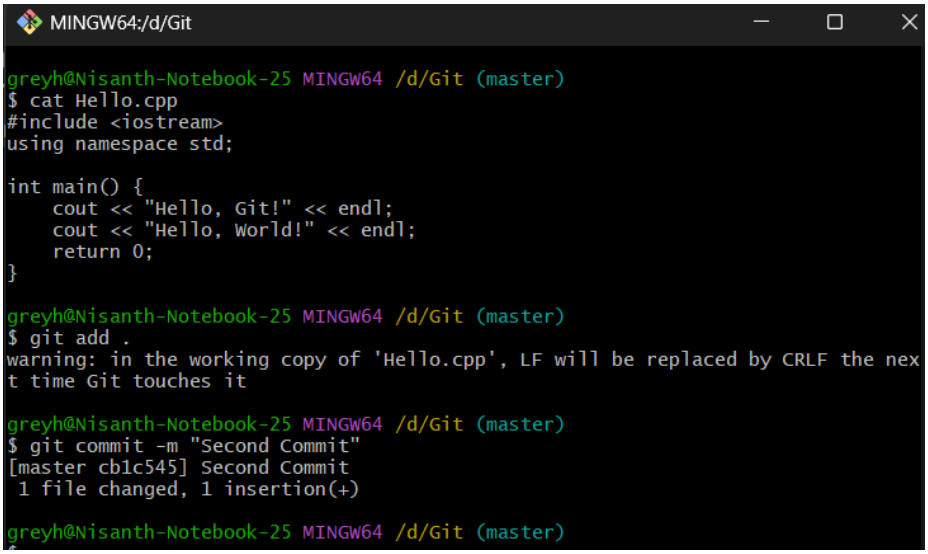


Figure – 4

Step 5: Modify the Hello.cpp File Again (Second Change)

- Make another change in the same Hello.cpp file.
- Example: Modify a different function or add a new comment.
- Save the file and commit it.

```
MINGW64:/d/Git
$ vi Hello.cpp

greyh@Nisanth-Notebook-25 MINGW64 /d/Git (master)
$ cat Hello.cpp
#include <iostream>
using namespace std;

int main() {
    cout << "Hello, Git!" << endl;
    cout << "Hello, World!" << endl;
    cout << "Hello, AI" << endl;
    return 0;
}

greyh@Nisanth-Notebook-25 MINGW64 /d/Git (master)
$ git add .
warning: in the working copy of 'Hello.cpp', LF will be replaced by CRLF the nex
t time Git touches it

greyh@Nisanth-Notebook-25 MINGW64 /d/Git (master)
$ git commit -m "Third Commit"
[master 22e17c8] Third Commit
 1 file changed, 1 insertion(+)
```

Figure – 5

Step 6: View Git Log Again in One Line Format

This will now show the latest two commits along with previous commits.

```
MINGW64:/d/Git
    cout << "Hello, Git!" << endl;
    cout << "Hello, World!" << endl;
    cout << "Hello, AI" << endl;
    return 0;
}

greyh@Nisanth-Notebook-25 MINGW64 /d/Git (master)
$ git add .
warning: in the working copy of 'Hello.cpp', LF will be replaced by CRLF the nex
t time Git touches it

greyh@Nisanth-Notebook-25 MINGW64 /d/Git (master)
$ git commit -m "Third Commit"
[master 22e17c8] Third Commit
 1 file changed, 1 insertion(+)

greyh@Nisanth-Notebook-25 MINGW64 /d/Git (master)
$ git log --oneline
22e17c8 (HEAD -> master) Third Commit
cb1c545 Second Commit
2c35438 Initial commit: Added main.cpp

greyh@Nisanth-Notebook-25 MINGW64 /d/Git (master)
$
```

Figure – 6

Step 7: View Differences Between Commits

The **git diff** command shows the exact lines changed between each commits. You can compare between multiple commits. Example: First commit and Second commit or Second commit and Third commit or even multiple commits.

This shows changes between the First commit and Second commit.

```
MINGW64:/d/Git
$ git log --oneline
22e17c8 (HEAD -> master) Third Commit
cb1c545 Second Commit
2c35438 Initial commit: Added main.cpp

greyh@Nisanth-Notebook-25 MINGW64 /d/Git (master)
$ git diff

greyh@Nisanth-Notebook-25 MINGW64 /d/Git (master)
$ git diff 2c35438 cb1c545
diff --git a/Hello.cpp b/Hello.cpp
index 4da7964..e50fbf8 100644
--- a/Hello.cpp
+++ b/Hello.cpp
@@ -3,5 +3,6 @@ using namespace std;

 int main() {
     cout << "Hello, Git!" << endl;
+    cout << "Hello, World!" << endl;
     return 0;
 }

greyh@Nisanth-Notebook-25 MINGW64 /d/Git (master)
```

Figure – 7

This shows changes between the Second commit and Third commit.

```
MINGW64:/d/Git
@@ -3,5 +3,6 @@ using namespace std;

 int main() {
     cout << "Hello, Git!" << endl;
+    cout << "Hello, World!" << endl;
     return 0;
 }

greyh@Nisanth-Notebook-25 MINGW64 /d/Git (master)
$ git diff cb1c545 22e17c8
diff --git a/Hello.cpp b/Hello.cpp
index e50fbf8..988dde0 100644
--- a/Hello.cpp
+++ b/Hello.cpp
@@ -4,5 +4,6 @@ using namespace std;
 int main() {
     cout << "Hello, Git!" << endl;
     cout << "Hello, World!" << endl;
+    cout << "Hello, AI" << endl;
     return 0;
 }

greyh@Nisanth-Notebook-25 MINGW64 /d/Git (master)
$
```

Figure – 8

Source Code Management

LAB REPORT – 4

Step 1: Sign in to GitHub

Open a web browser and go to github.com

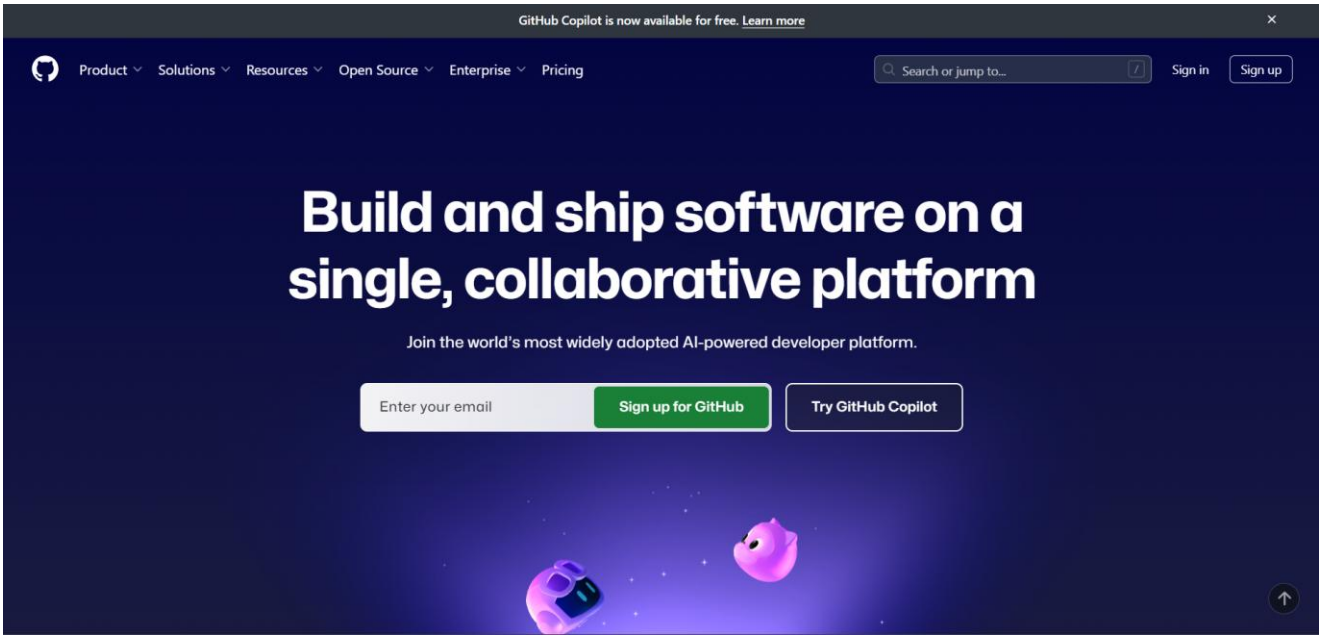


Figure – 1

Click Sign in and enter your credentials.

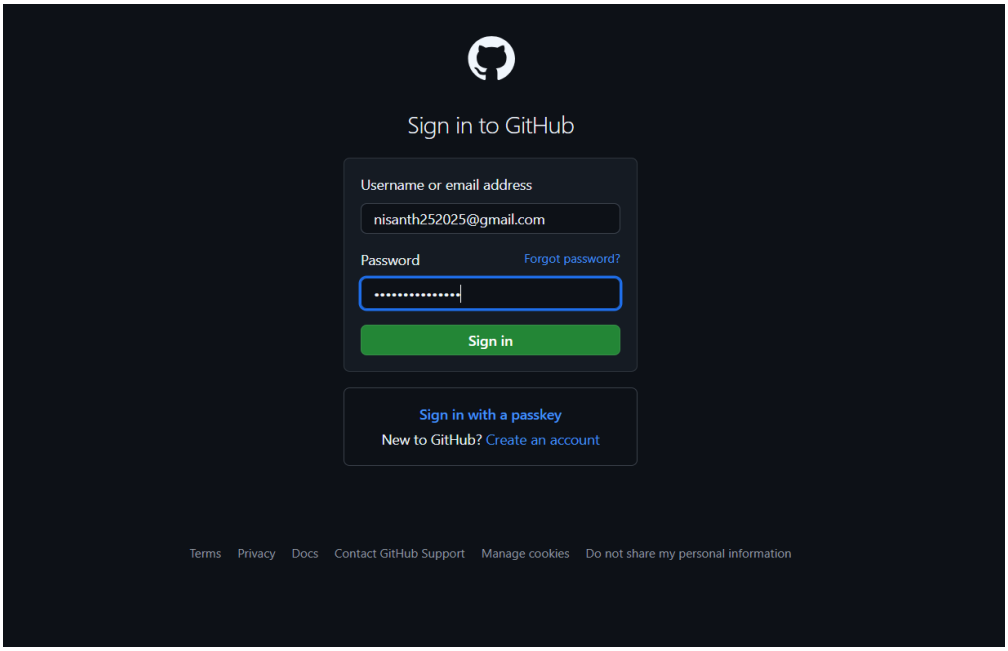


Figure – 2

Step 2: Creating a Repository

Click on the **"+"** icon (top-right corner) and select "**New repository**".

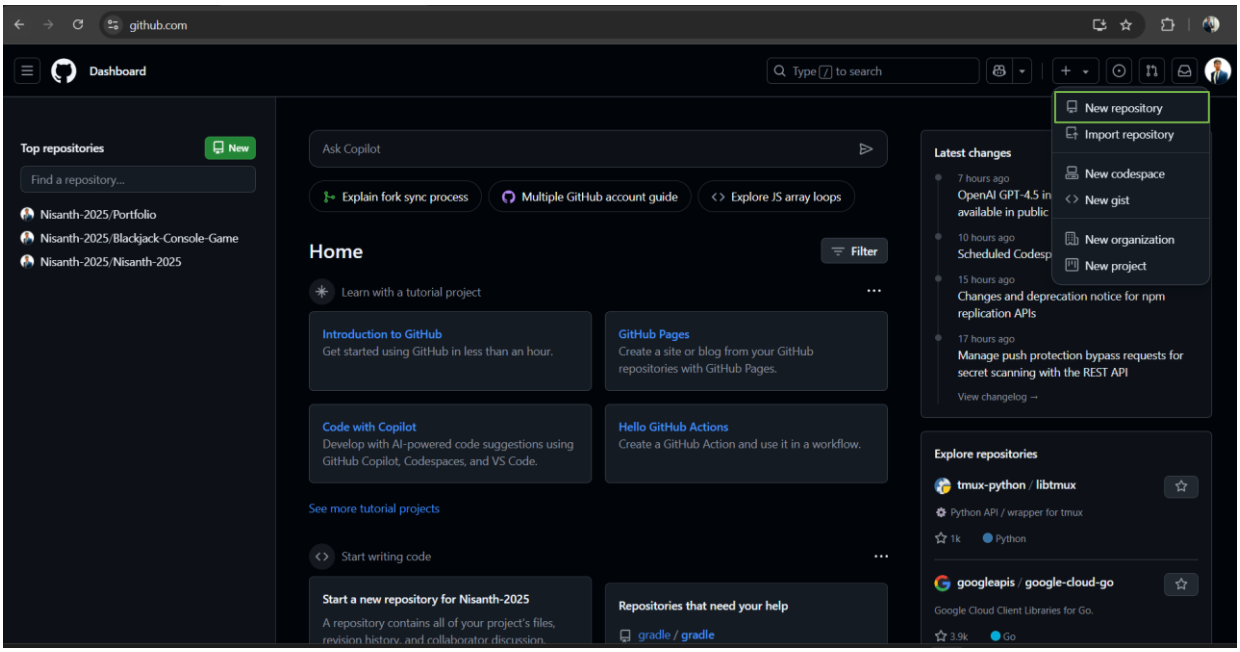


Figure – 3

In the **Repository name** field, enter the same name as your local folder. Select Public. **Do not** check "**Initialize this repository with a README**". Click **Create repository**.

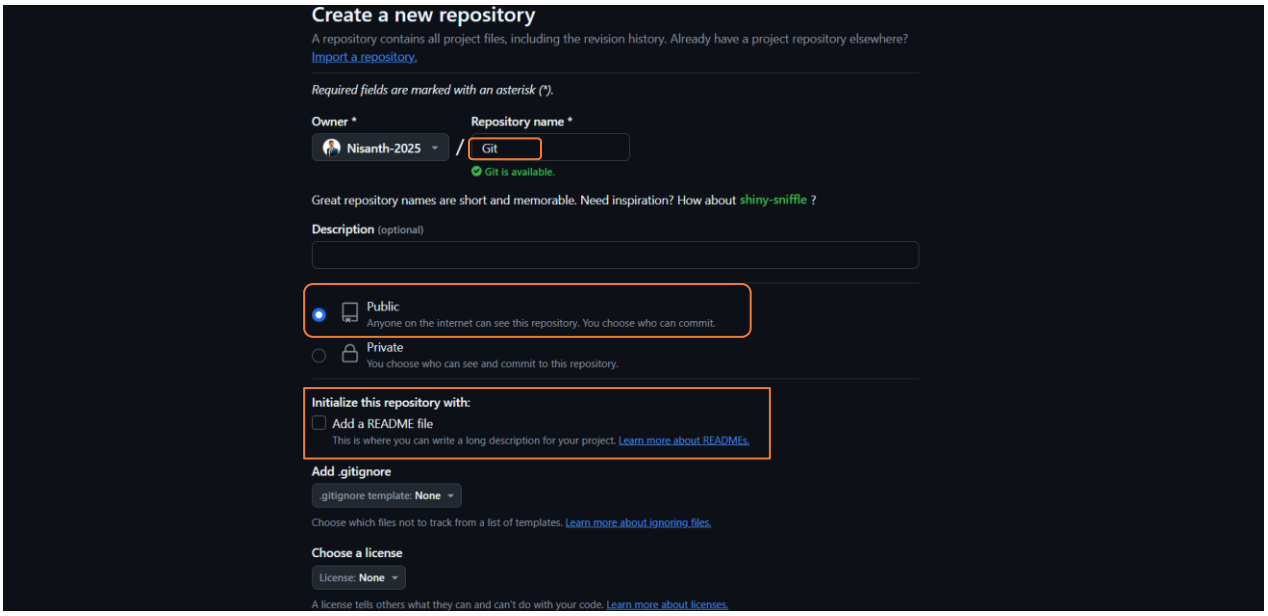


Figure – 4

Step 3: Connect Local Repository to GitHub

On the next page, copy the **HTTPS URL** under **"Quick setup"** it looks like

(**https://github.com/yourusername/repositoryname.git**).

Add the GitHub repository as a remote:

- **git remote**
- **git remote add origin <repository-URL>**

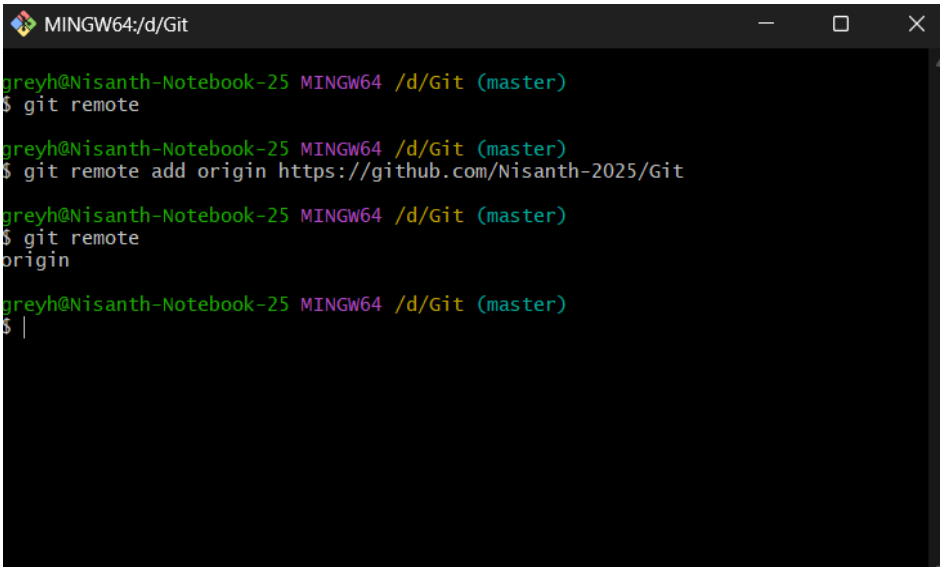


Figure – 5

Step 4: Push Code To GitHub

Push the committed files to GitHub using the command: **git push -u origin master**

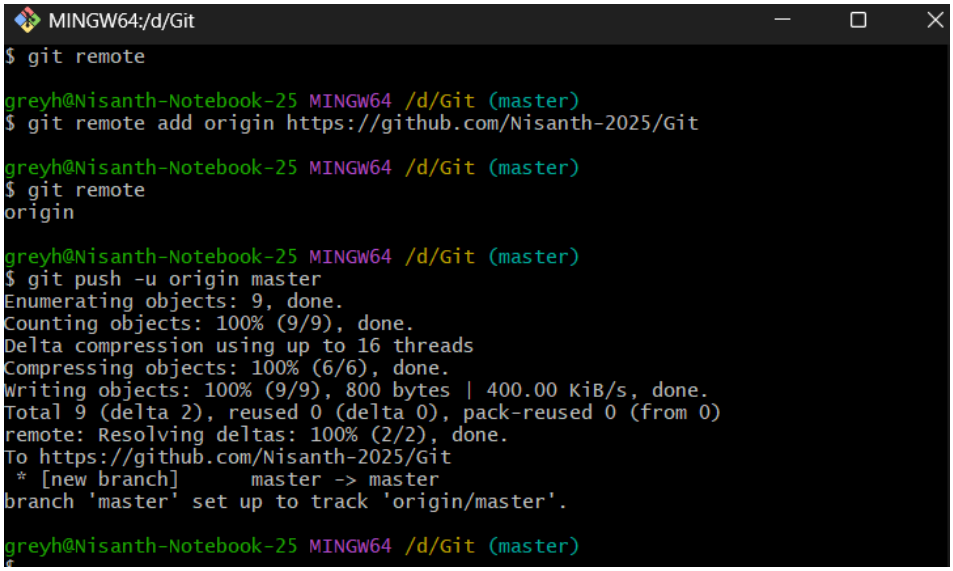


Figure – 6

Step 5: Verify Changes on GitHub

1. Open **GitHub** in your browser.
2. Go to your repository.
3. Refresh the page – your files should be visible in the repository.

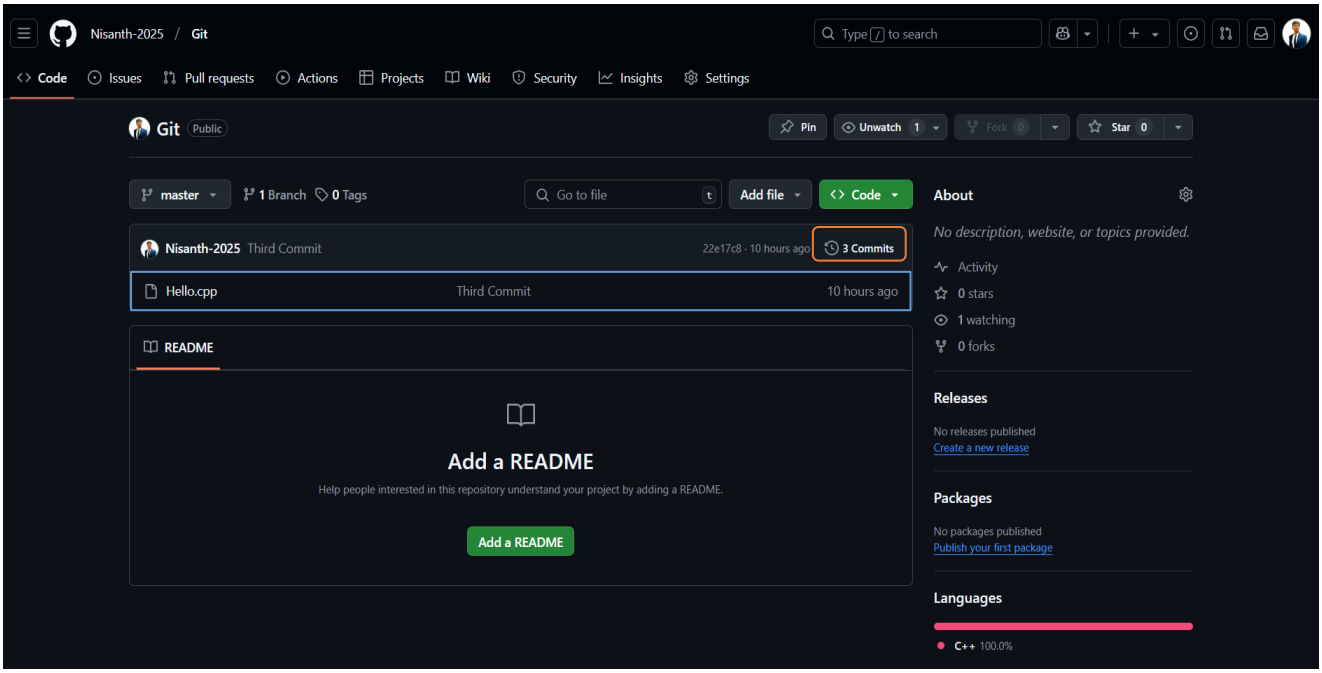


Figure – 7

Step 6: Edit the File Directly on GitHub

1. Click on Hello.cpp file in your GitHub repository.

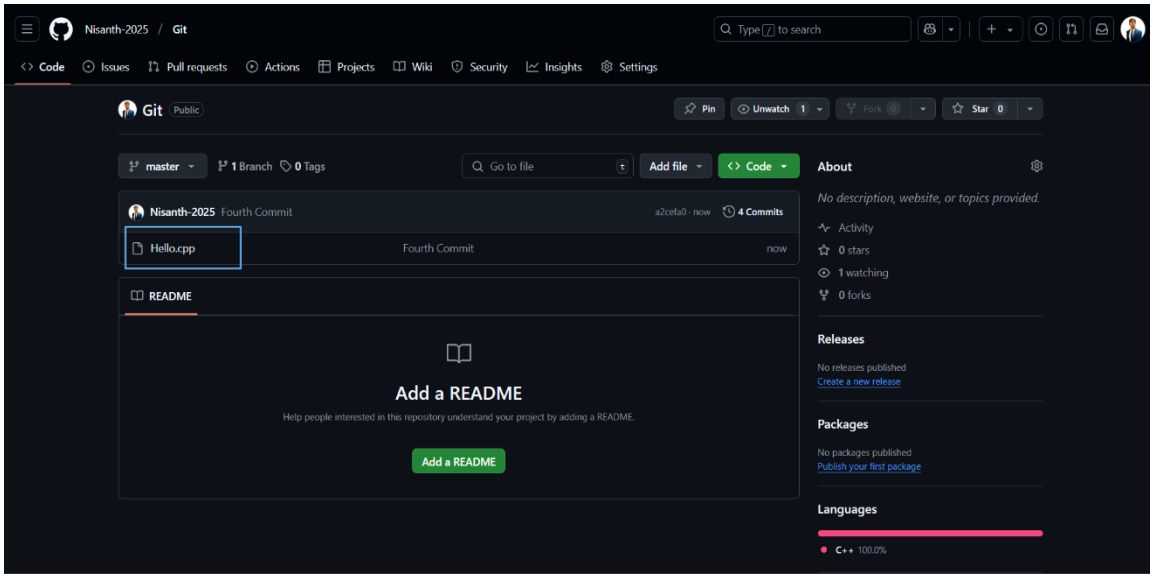


Figure – 8

2. Click the edit (pencil) icon in the top-right.

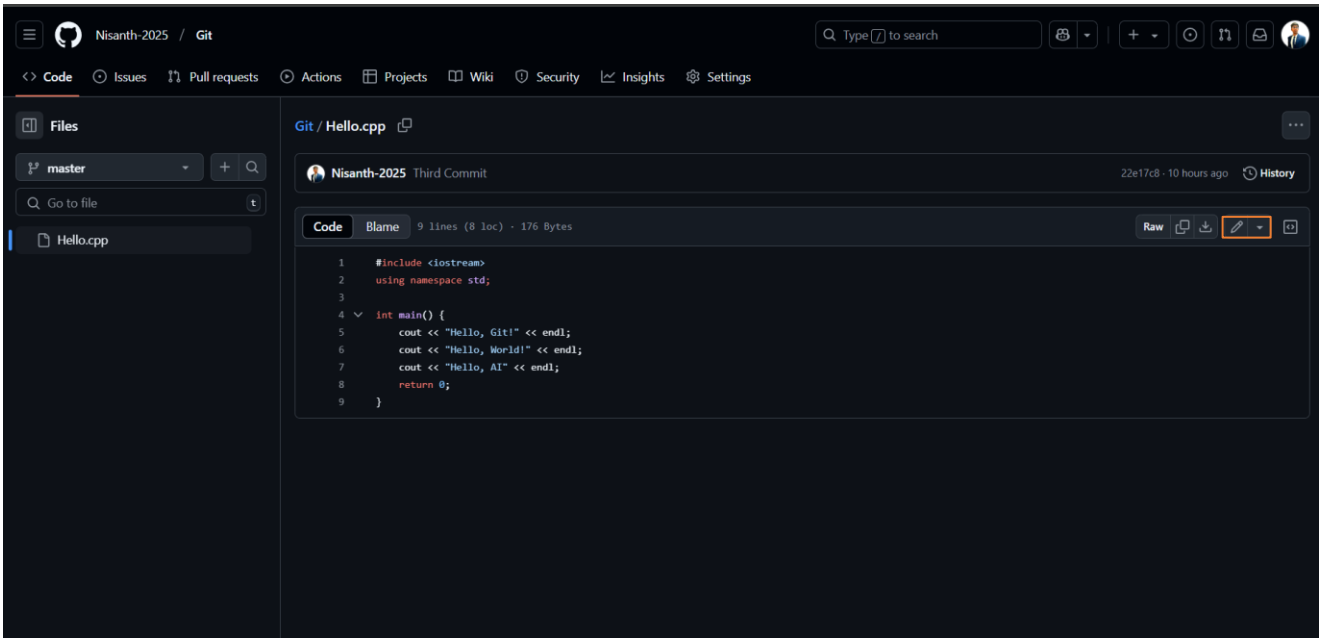


Figure – 9

3. Make some changes to the file, scroll down, enter a commit message, and click Commit changes.

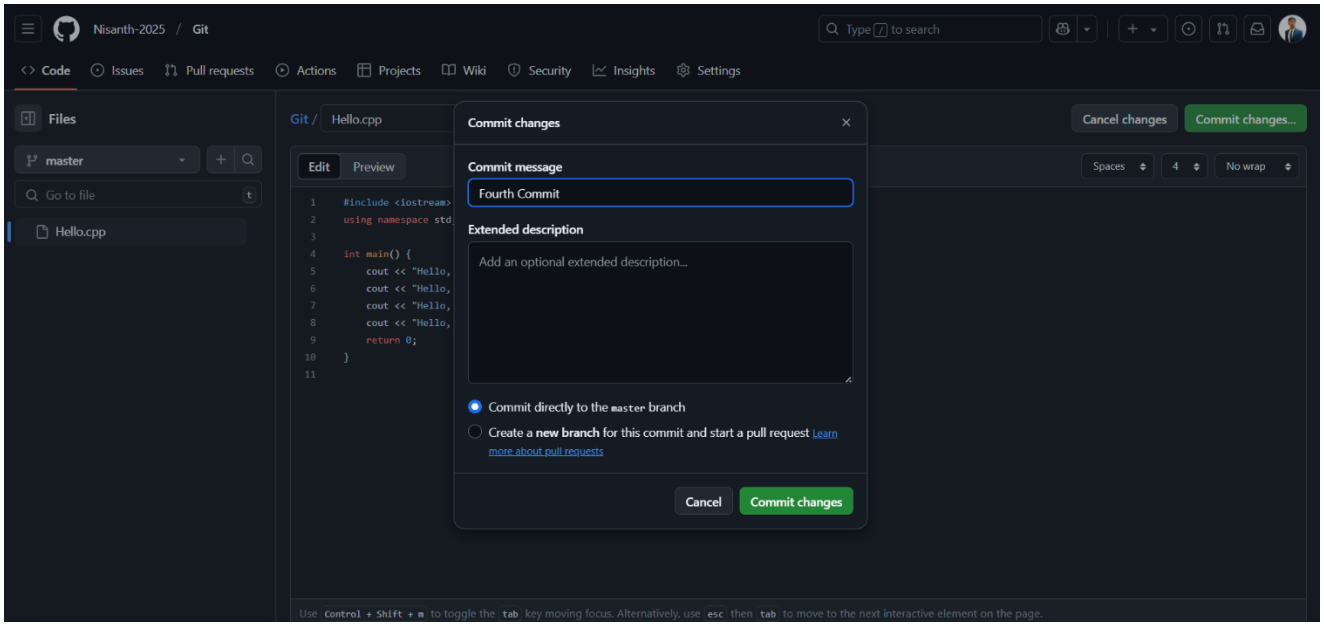


Figure – 10

Step 7: Pull Changes from GitHub to Local System

Open **Git Bash** in your project folder and Pull the latest changes from GitHub using the command: **git pull**

The updated file will now be available on your local system.

```
MINGW64:/d/Git
Delta compression using up to 16 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (9/9), 800 bytes | 400.00 KiB/s, done.
Total 9 (delta 2), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (2/2), done.
To https://github.com/Nisanth-2025/Git
 * [new branch]      master -> master
branch 'master' set up to track 'origin/master'.

greyh@Nisanth-Notebook-25 MINGW64 /d/Git (master)
$ git pull
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
Unpacking objects: 100% (3/3), 943 bytes | 117.00 KiB/s, done.
From https://github.com/Nisanth-2025/Git
   22e17c8..a2cefa0  master     -> origin/master
Updating 22e17c8..a2cefa0
Fast-forward
 Hello.cpp | 1 +
 1 file changed, 1 insertion(+)

greyh@Nisanth-Notebook-25 MINGW64 /d/Git (master)
```

Figure – 11

Use git log to see the changes in your local repository file.

```
MINGW64:/d/Git

greyh@Nisanth-Notebook-25 MINGW64 /d/Git (master)
$ git log
commit a2cefa01779d2fcf313d1785e888caa99ba79772 (HEAD -> master, origin/master,
origin/HEAD)
Author: Nisanth S <nisanth252025@gmail.com>
Date:   Fri Feb 28 22:31:21 2025 +0530

    Fourth Commit

commit 22e17c869df88622f51038b08390c694c3383412
Author: Nisanth-2025 <nisanth252025@gmail.com>
Date:   Fri Feb 28 12:10:56 2025 +0530

    Third Commit

commit cb1c54595497b090f7a38e80ba1eb4d9518f5c8d
Author: Nisanth-2025 <nisanth252025@gmail.com>
Date:   Fri Feb 28 12:05:17 2025 +0530

    Second Commit

commit 2c354383229249ecb538ad183981698d73313dc7
Author: Nisanth-2025 <nisanth252025@gmail.com>
```

Figure – 12

Source Code Management

LAB REPORT – 5

Step 1: Create a New Branch

Use the following command to create a new branch named **dev** and switch to it:

```
git checkout -b dev
```

```
nisan@Nisanth-NoteBook-25 MINGW64 /d/Git (master)
$ git commit -m "first commit"
[master (root-commit) 93df596] first commit
 1 file changed, 10 insertions(+)
 create mode 100644 hello.cpp

nisan@Nisanth-NoteBook-25 MINGW64 /d/Git (master)
$ git checkout -b dev
Switched to a new branch 'dev'

nisan@Nisanth-NoteBook-25 MINGW64 /d/Git (dev)
$
```

Figure - 1

Step 2: Make Changes in the dev Branch

Open the **hello.cpp** file and make some changes.

```
        hello.cpp

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

nisan@Nisanth-NoteBook-25 MINGW64 /d/Git (master)
$ git add .
warning: in the working copy of 'hello.cpp', LF will be replaced by C
g
nisan@Nisanth-NoteBook-25 MINGW64 /d/Git (master)
$ git commit -m "first commit"
[master (root-commit) 93df596] first commit
 1 file changed, 10 insertions(+)
 create mode 100644 hello.cpp

nisan@Nisanth-NoteBook-25 MINGW64 /d/Git (master)
$ git checkout -b dev
Switched to a new branch 'dev'

nisan@Nisanth-NoteBook-25 MINGW64 /d/Git (dev)
$ vi hello.cpp

nisan@Nisanth-NoteBook-25 MINGW64 /d/Git (dev)
$
```

Figure – 2

Step 3: Stage and Commit Changes

- **git add .**
- **git commit -m "Added a new file in dev branch"**

```
nisan@Nisanth-NoteBook-25 MINGW64 /d/Git (master)
$ git checkout -b dev
Switched to a new branch 'dev'

nisan@Nisanth-NoteBook-25 MINGW64 /d/Git (dev)
$ vi hello.cpp

nisan@Nisanth-NoteBook-25 MINGW64 /d/Git (dev)
$ git add .
git commit -m "Added a new file in dev branch"
warning: in the working copy of 'hello.cpp', LF will be replaced by CRLF the next time Git touches i
[dev 9892e31] Added a new file in dev branch
 1 file changed, 1 insertion(+)

nisan@Nisanth-NoteBook-25 MINGW64 /d/Git (dev)
$ git status
On branch dev
nothing to commit, working tree clean

nisan@Nisanth-NoteBook-25 MINGW64 /d/Git (dev)
```

Figure – 3

Step 4: Switch Back to master Branch

git checkout master

```
nisan@Nisanth-NoteBook-25 MINGW64 /d/Git (dev)
$ vi hello.cpp

nisan@Nisanth-NoteBook-25 MINGW64 /d/Git (dev)
$ git add .
git commit -m "Added a new file in dev branch"
warning: in the working copy of 'hello.cpp', LF will be replaced by CRLF the next time Git
[dev 9892e31] Added a new file in dev branch
 1 file changed, 1 insertion(+)

nisan@Nisanth-NoteBook-25 MINGW64 /d/Git (dev)
$ git status
On branch dev
nothing to commit, working tree clean

nisan@Nisanth-NoteBook-25 MINGW64 /d/Git (dev)
$ git checkout master
Switched to branch 'master'

nisan@Nisanth-NoteBook-25 MINGW64 /d/Git (master)
$
```

Figure – 4

Step 5: Merge dev into master

If there are no conflicts, this will merge the changes from the dev branch into master.

git merge dev

```
nisan@Nisanth-NoteBook-25 MINGW64 /d/Git (dev)
$ git status
On branch dev
nothing to commit, working tree clean

nisan@Nisanth-NoteBook-25 MINGW64 /d/Git (dev)
$ git checkout master
Switched to branch 'master'

nisan@Nisanth-NoteBook-25 MINGW64 /d/Git (master)
$ git merge dev
Updating 93df596..9892e31
Fast-forward
 hello.cpp | 1 +
 1 file changed, 1 insertion(+)

nisan@Nisanth-NoteBook-25 MINGW64 /d/Git (master)
$ |
```

Figure - 5

Step 6: Verify the Merge

Use cat command to check is the files are merged.

cat hello.cpp

```
nisan@Nisanth-NoteBook-25 MINGW64 /d/Git (master)
$ ls
hello.cpp

nisan@Nisanth-NoteBook-25 MINGW64 /d/Git (master)
$ cat hello.cpp
#include <iostream>
using namepsace std;
int main()
{
        int num1,num2,sum;
        cin >> num1;
        cin >> num2;
        sum = num1 + num2;
        cout< sum;
        cout<< "Hello to dev branch";
}

nisan@Nisanth-NoteBook-25 MINGW64 /d/Git (master)
$
```

Figure – 6

Step 7: Run the Git Merge Tool

Use git mergetool to open the conflict screen. Close it using **escape :wqa**

```
git mergetool
```

Figure – 7

Step 8: Creating a .gitignore File

The **.gitignore** file tells Git to ignore specific files or directories that do not need to be tracked, such as log files, build directories, or system files.

```
touch .gitignore
```

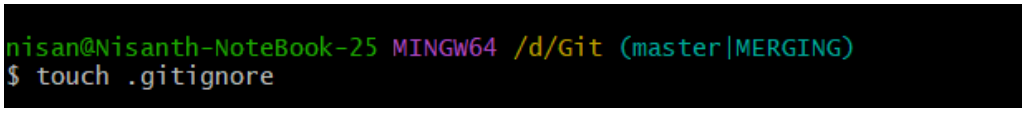


Figure – 8

Step 9: Viewing Hidden Files and Folders

By default, files that start with a dot (.) are **hidden** in Unix-based systems, including Git Bash.

```
ls -a
```

```
nisan@Nisanth-NoteBook-25 MINGW64 /d/Git (master|MERGING)
$ touch .gitignore

nisan@Nisanth-NoteBook-25 MINGW64 /d/Git (master|MERGING)
$ ls -a
./  ../  .git/  .gitignore  hi.txt  hi.txt.orig

nisan@Nisanth-NoteBook-25 MINGW64 /d/Git (master|MERGING)
$ |
```

Figure – 8

Source Code Management

LAB REPORT – 6

Step 1: Fork a Repository on GitHub

- Go to any public repository on GitHub (e.g., https://github.com/octocat/Hello-World).
- Click on the **"Fork"** button (top right corner).
- This creates a copy of the repository under **your GitHub account**.

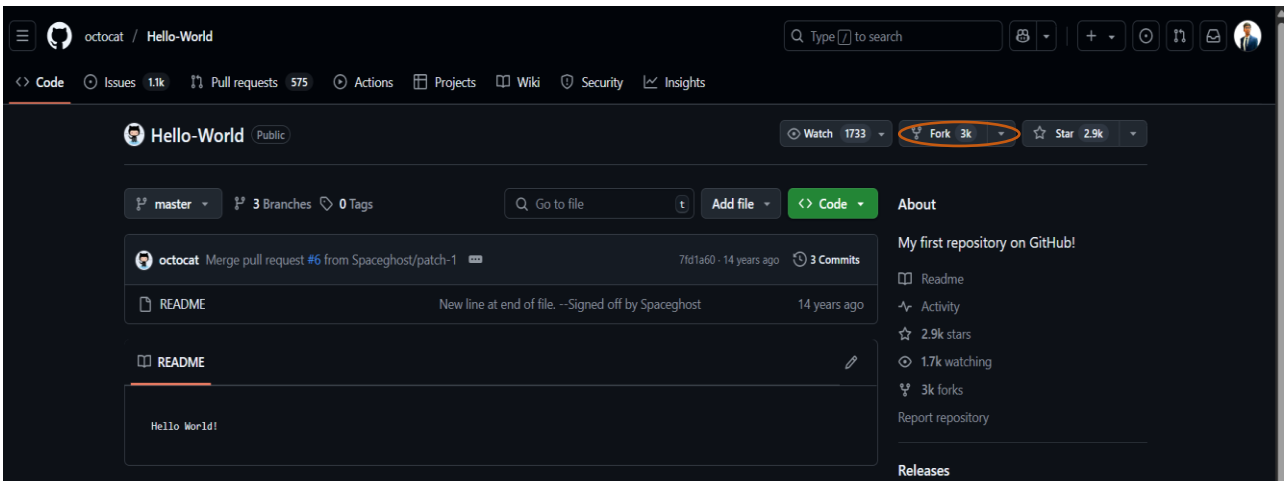


Figure - 1

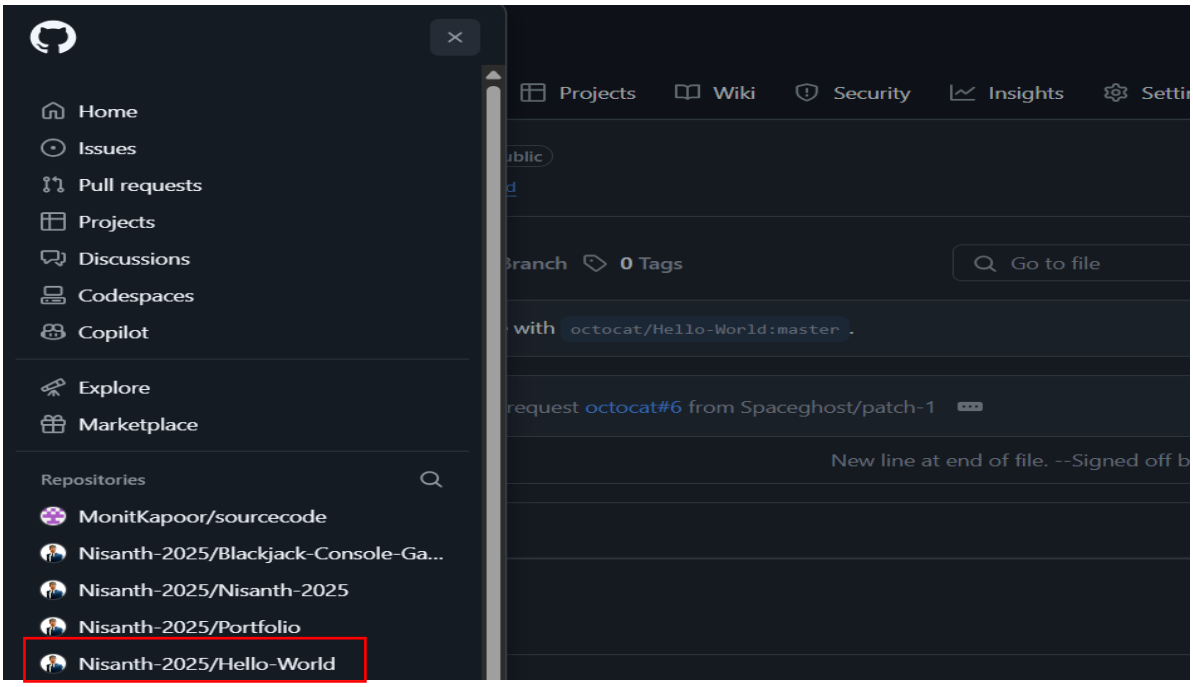


Figure – 2

Step 2: Clone the Forked Repository Locally

Replace your-**username** with your actual GitHub username.

```
git clone https://github.com/your-username/Hello-World.git
```

```
nisan@Nisanth-NoteBook-25 MINGW64 /d/Git (master|MERGING)
$ git clone https://github.com/Nisanth-2025/Hello-World
Cloning into 'Hello-World'...
remote: Enumerating objects: 7, done.
remote: Total 7 (delta 0), reused 0 (delta 0), pack-reused 7 (from 1)
Receiving objects: 100% (7/7), done.

nisan@Nisanth-NoteBook-25 MINGW64 /d/Git (master|MERGING)
$
```

Figure – 3

Step 3: Change Directory to the Cloned Repo

```
cd Hello-World
```

```
nisan@Nisanth-NoteBook-25 MINGW64 /d/Git (master|MERGING)
$ cd Hello-World

nisan@Nisanth-NoteBook-25 MINGW64 /d/Git/Hello-World (master)
$
```

Figure – 4

Step 4: Add a New File or Modify Existing One

```
Vi hello.cpp
```

```
nisan@Nisanth-NoteBook-25 MINGW64 /d/Git (master|MERGING)
$ cd Hello-World

nisan@Nisanth-NoteBook-25 MINGW64 /d/Git/Hello-World (master)
$ ls
README

nisan@Nisanth-NoteBook-25 MINGW64 /d/Git/Hello-World (master)
$ vi hello.cpp

nisan@Nisanth-NoteBook-25 MINGW64 /d/Git/Hello-World (master)
$ |
```

Figure – 5

Step 5: Stage and Commit Your Changes

```
git add .

git commit -m "First Commit"
```

```
nisan@Nisanth-NoteBook-25 MINGW64 /d/Git/Hello-World (master)
$ git add .
warning: in the working copy of 'hello.cpp', LF will be replaced by CRLF the next time Git touches it

nisan@Nisanth-NoteBook-25 MINGW64 /d/Git/Hello-World (master)
$  git commit -m "first commit"
[master 3b7dc11] first commit
 1 file changed, 11 insertions(+)
 create mode 100644 hello.cpp

nisan@Nisanth-NoteBook-25 MINGW64 /d/Git/Hello-World (master)
$ |
```

Figure – 6

Step 6: Push Changes to Your Forked GitHub Repo

This updates **your forked repository** on GitHub with your changes.

```
git push origin master
```

```
nisan@Nisanth-NoteBook-25 MINGW64 /d/Git/Hello-World (master)
$ git push origin master
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 16 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 383 bytes | 383.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/Nisanth-2025/Hello-World
   7fd1a60..3b7dc11  master -> master

nisan@Nisanth-NoteBook-25 MINGW64 /d/Git/Hello-World (master)
$ |
```

Figure – 7

Step 7: Create a Pull Request

If you want your changes to be added to the original repository:

1. Go to your forked repo on GitHub.
2. Click **"Contribute" > "Open Pull Request"**.
3. Submit your pull request for review.

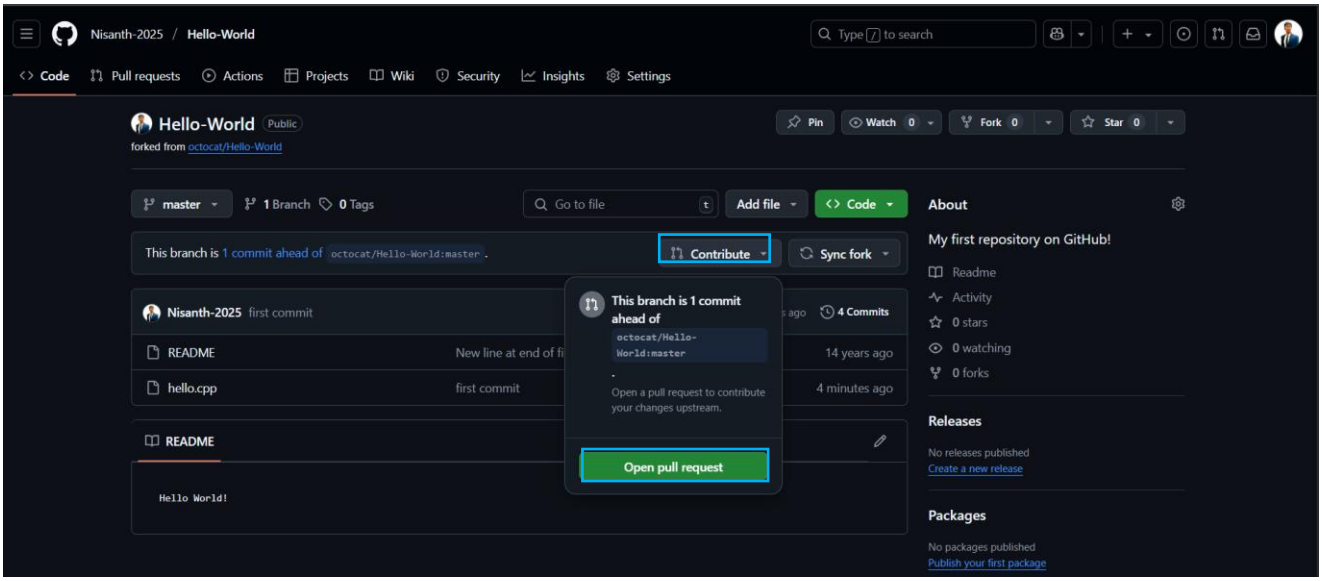
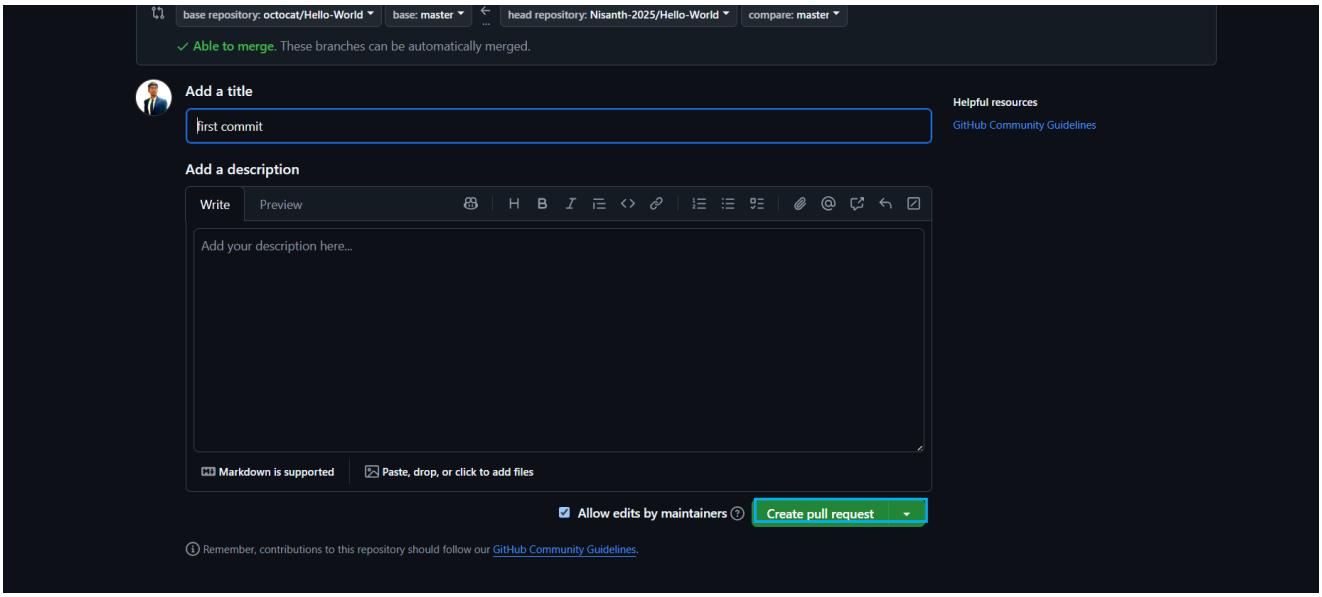

Nisanth-2025 / Hello-World
Type / to search
Code
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
Hello-World
Public
forked from octocat/Hello-World
Pin
Watch 0
Fork 0
Star 0
master
1 Branch
0 Tags
Go to file
Add file
Code
This branch is 1 commit ahead of octocat/Hello-World:master .
Contribute
Sync fork
Nisanth-2025 first commit
4 Commits
README
New line at end of fi
14 years ago
hello.cpp
first commit
4 minutes ago
README
Hello World!
This branch is 1 commit ahead of
octocat/Hello-World:master
Open a pull request to contribute your changes upstream.
Open pull request
About
My first repository on GitHub!
Readme
Activity
0 stars
0 watching
0 forks
Releases
No releases published
Create a new release
Packages
No packages published
Publish your first package

Figure - 8


base repository: octocat/Hello-World
base: master
head repository: Nisanth-2025/Hello-World
compare: master
Able to merge. These branches can be automatically merged.
Add a title
first commit
Helpful resources
GitHub Community Guidelines
Add a description
Write
Preview
Add your description here...
Markdown is supported
Paste, drop, or click to add files
Allow edits by maintainers
Create pull request
Remember, contributions to this repository should follow our GitHub Community Guidelines.