Source Code Management

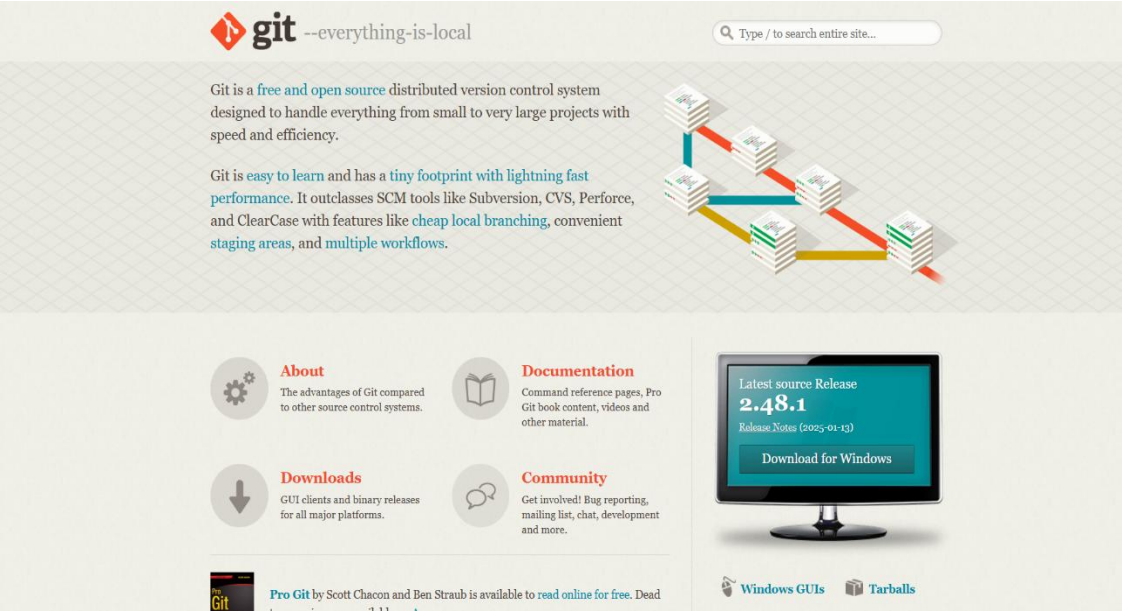
LAB REPORT – 1

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.

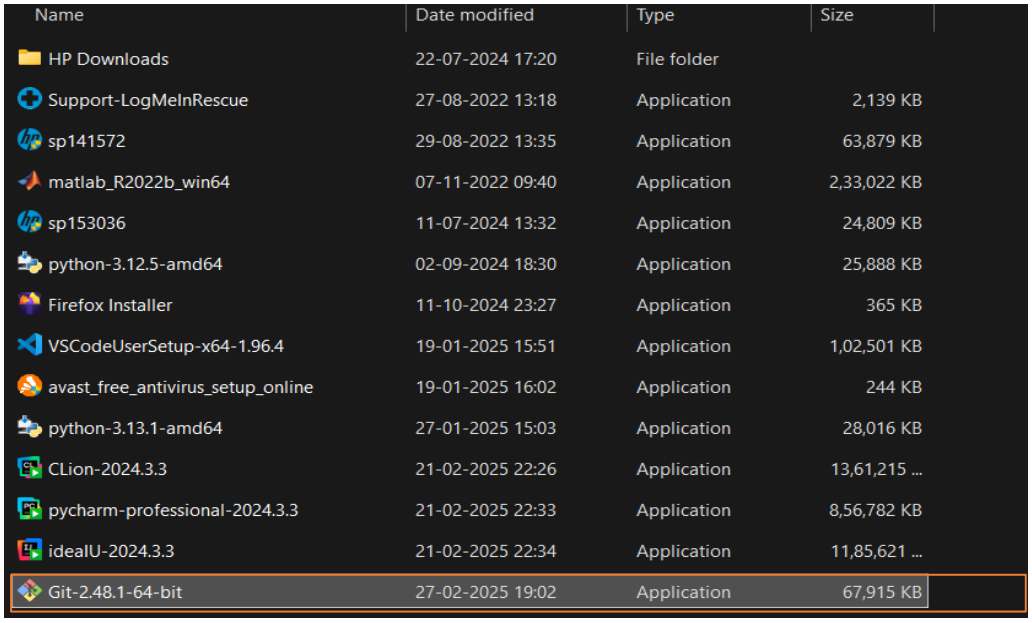
Step 1: Downloading Git

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

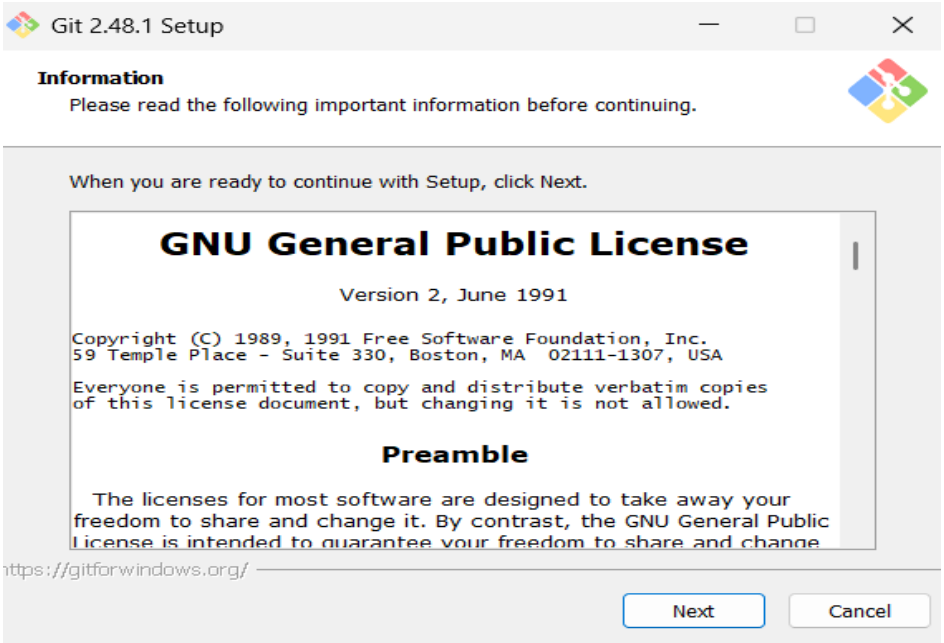
Locate the downloaded **Git.exe** file and double-click to run it.



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

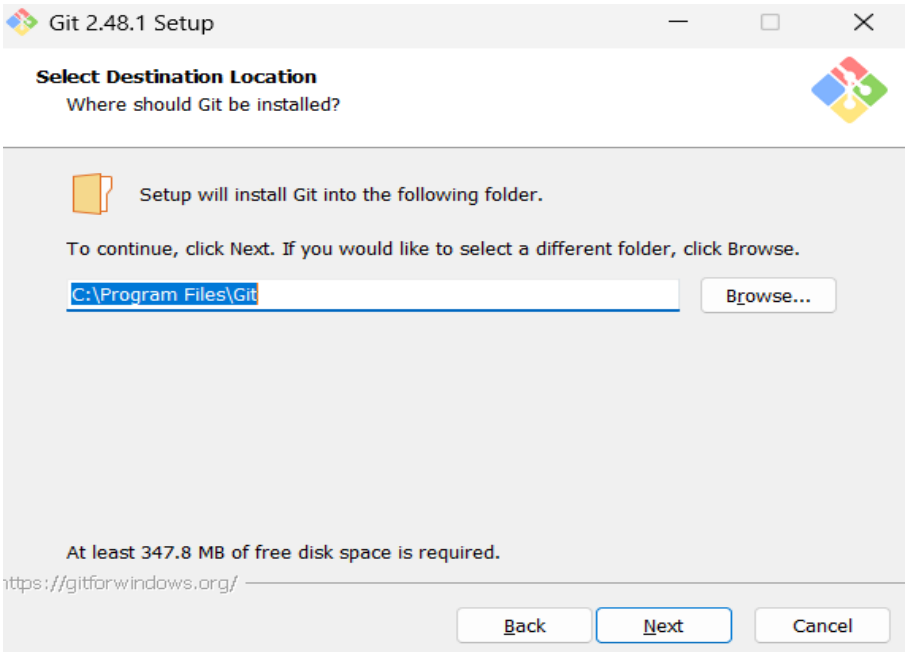
Step 3: License (Terms and Conditions)

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



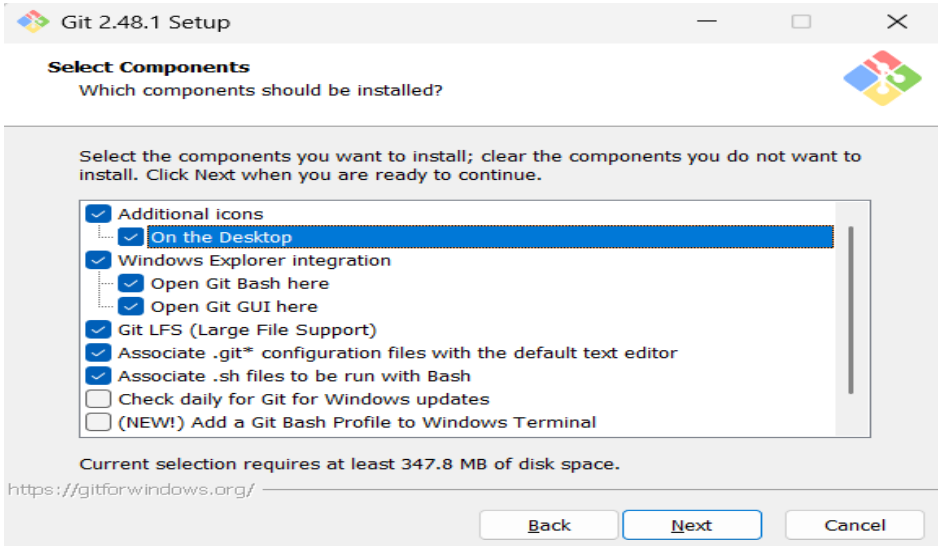
Step 4: Choose Installation Location

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



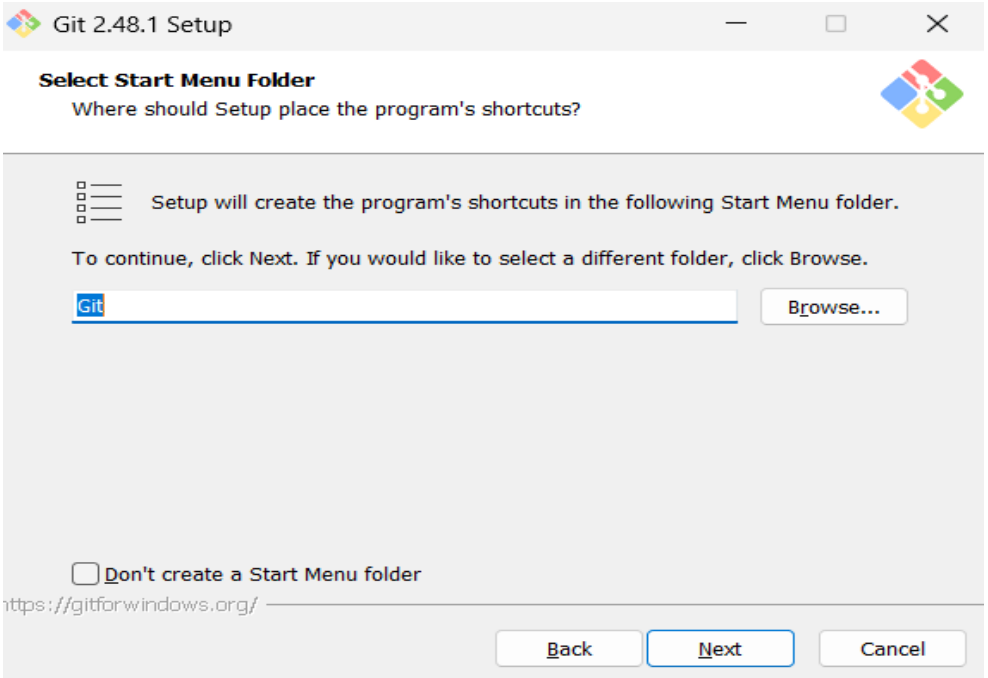
Step 5: Select the Components

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



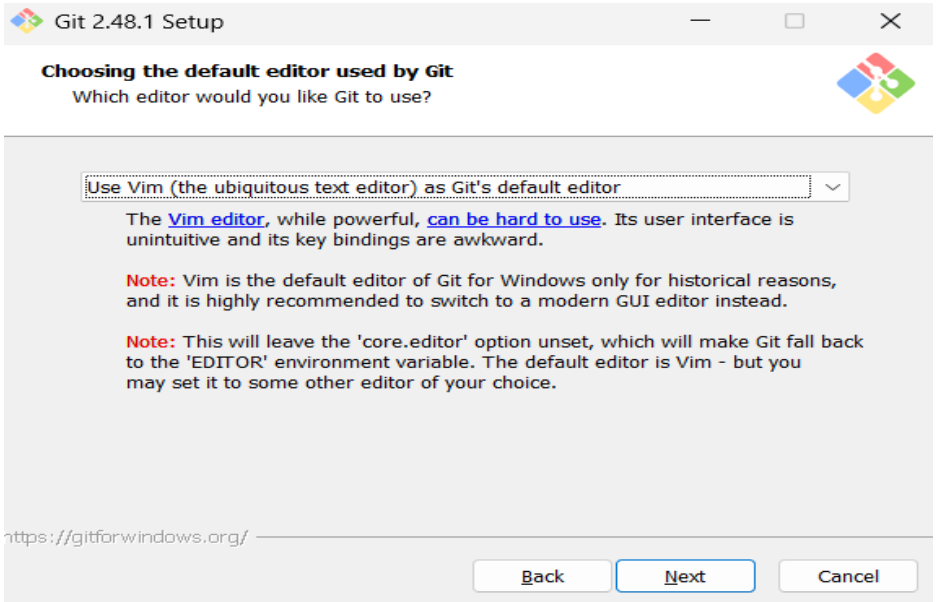
Step 6: Select Start Menu Folder

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.



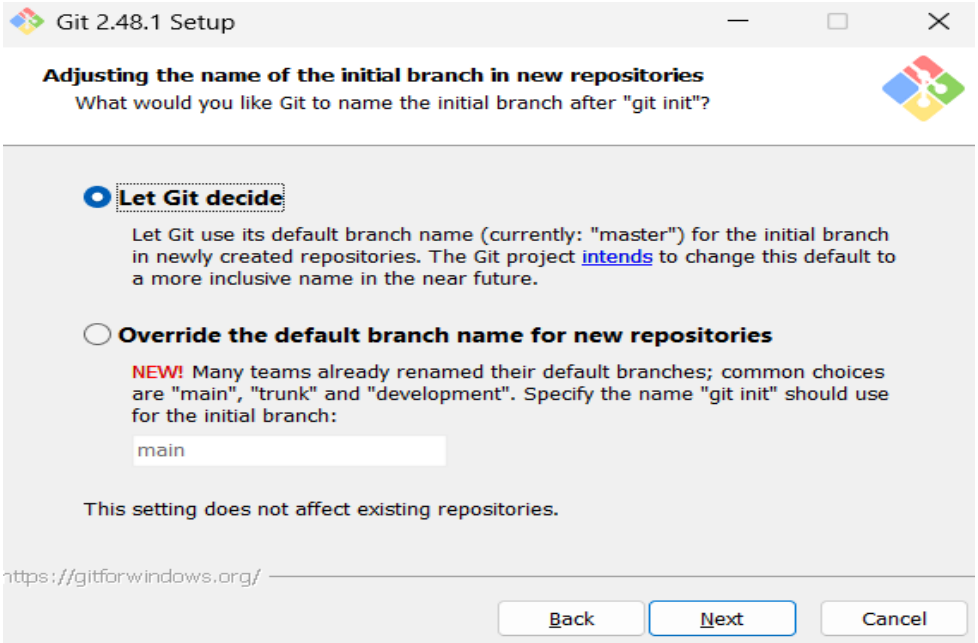
Step 7: Choose the Text Editor

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



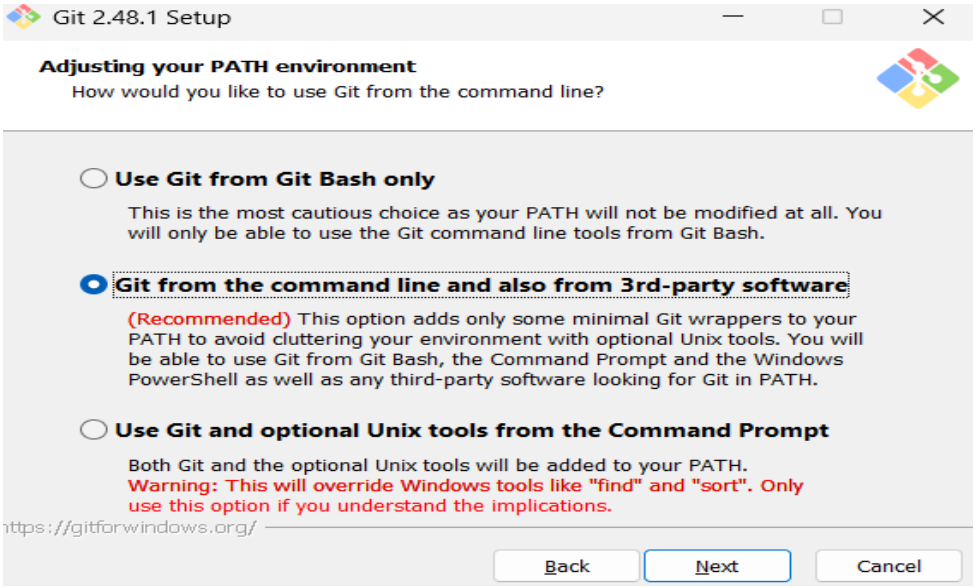
Step 8: Adjusting Initial Branch Name

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**.



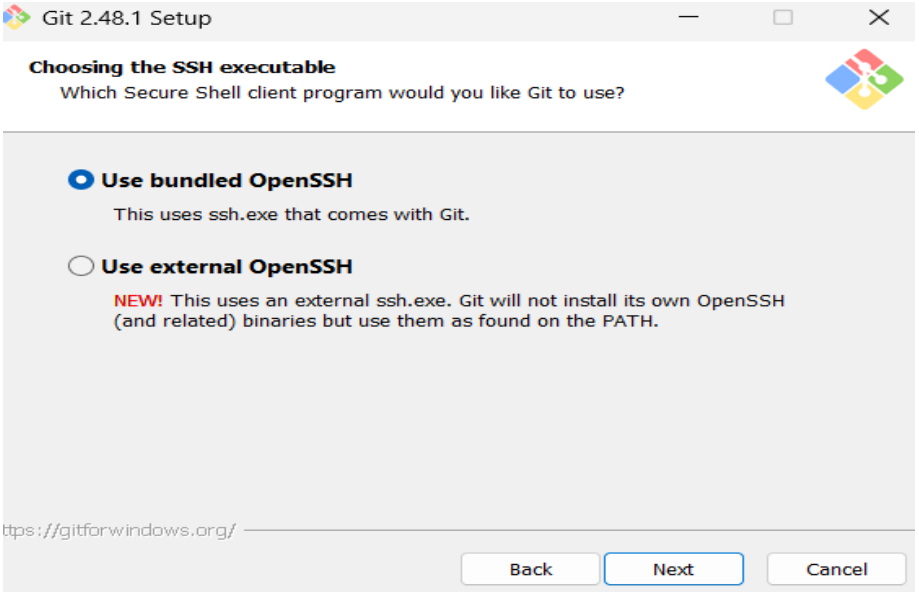
Step 9: Adjusting PATH Environment

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



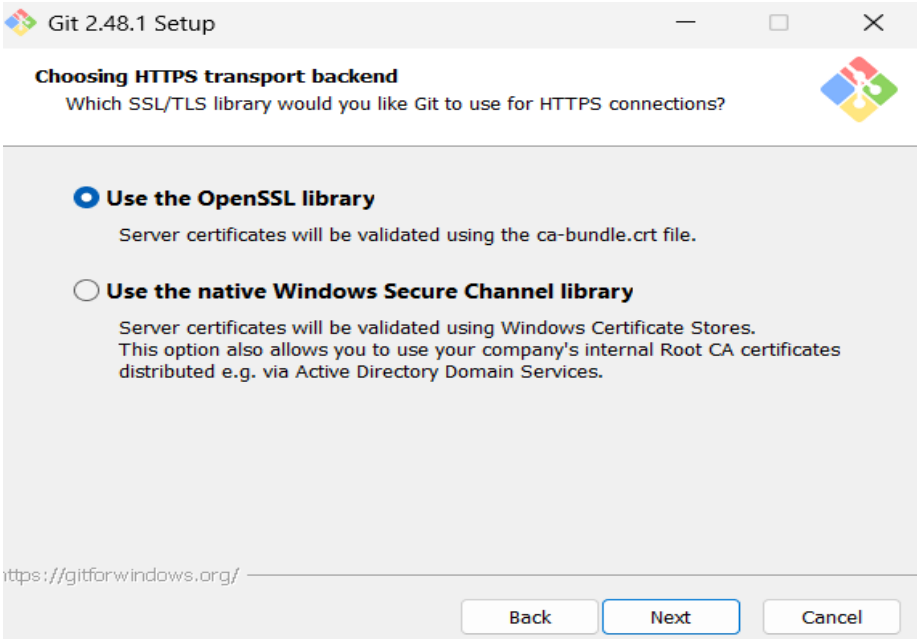
Step 10: Choosing the SSH Executable

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



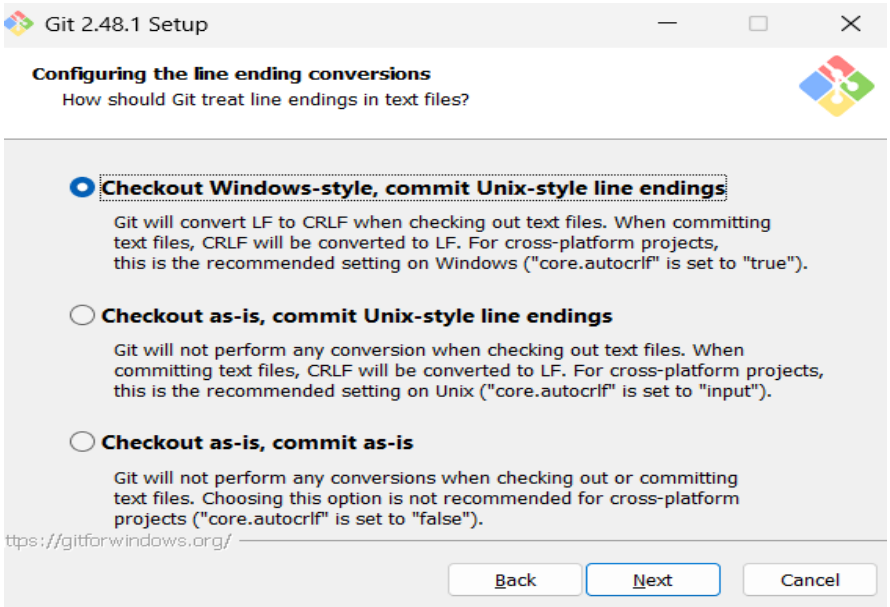
Step 11: Choosing the HTTP Transport Background

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



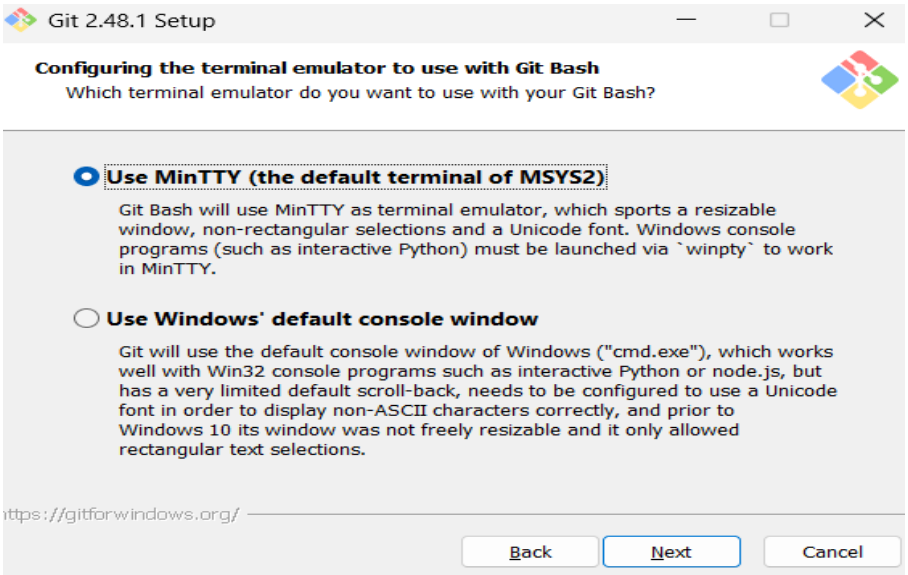
Step 12: Configuring Line Ending Configs

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



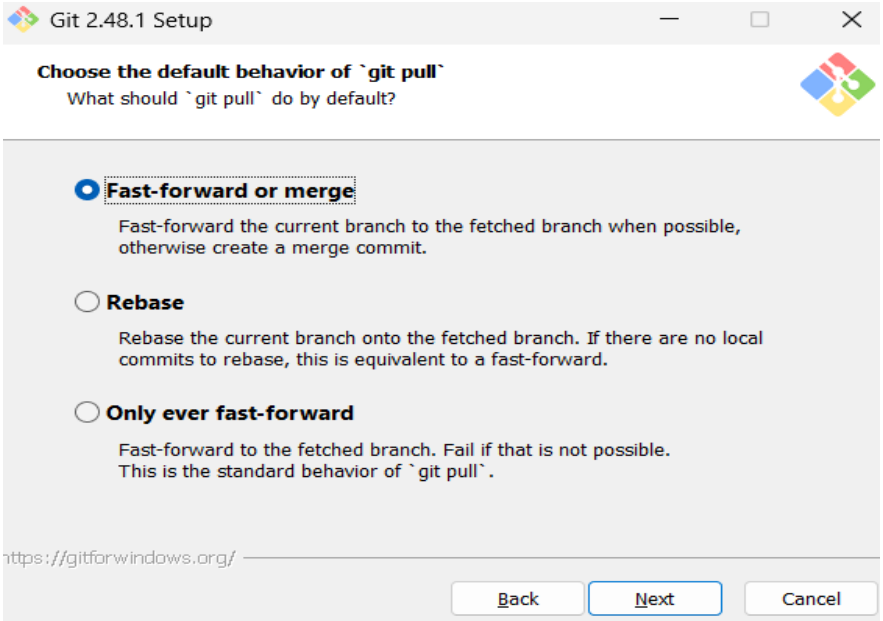
Step 13: Configuring the Terminal Emulator

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



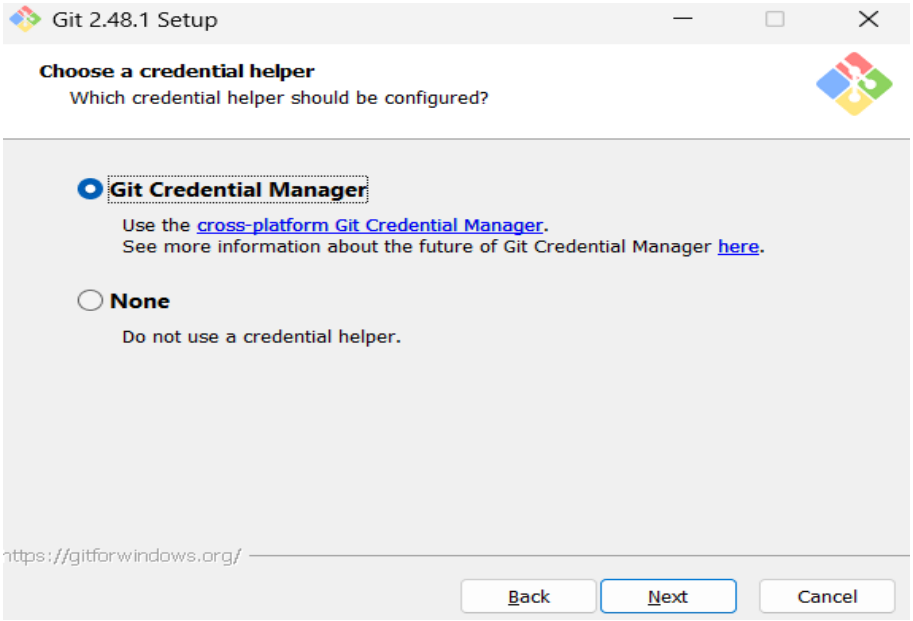
Step 14: Choosing the Default Behaviour

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



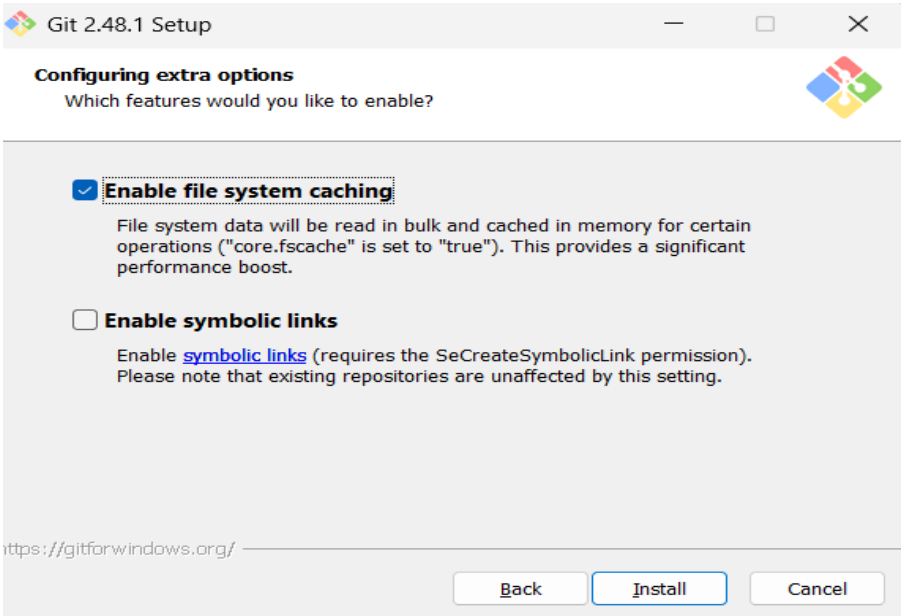
Step 15: Choosing a Credential Helper

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



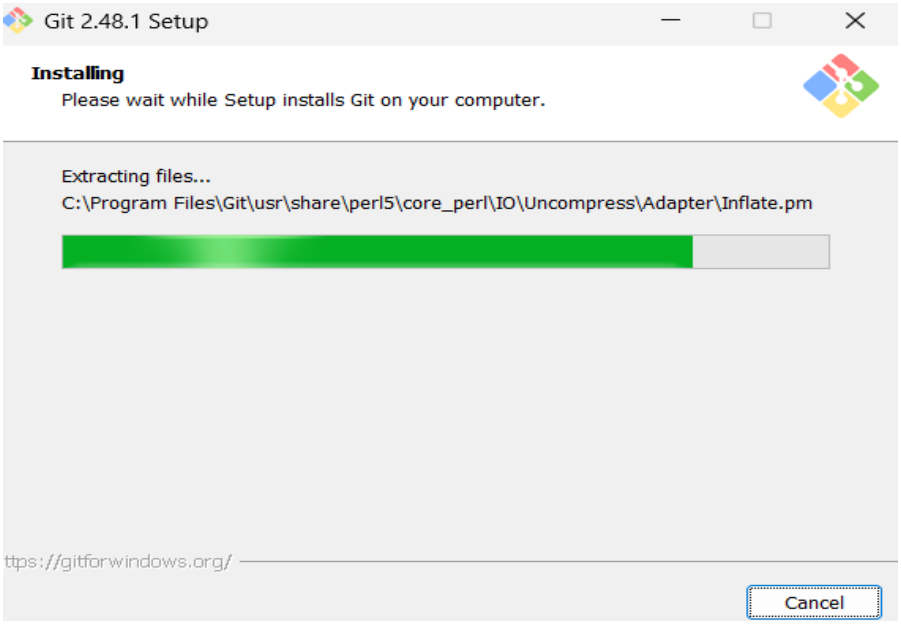
Step 16: Configuring Extra Options

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



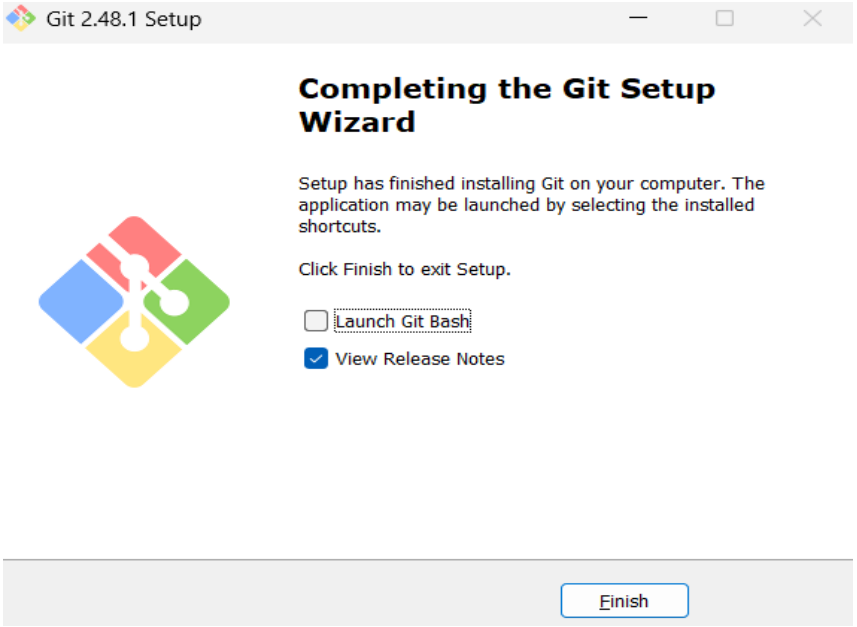
Step 17: Installation Overview

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**.

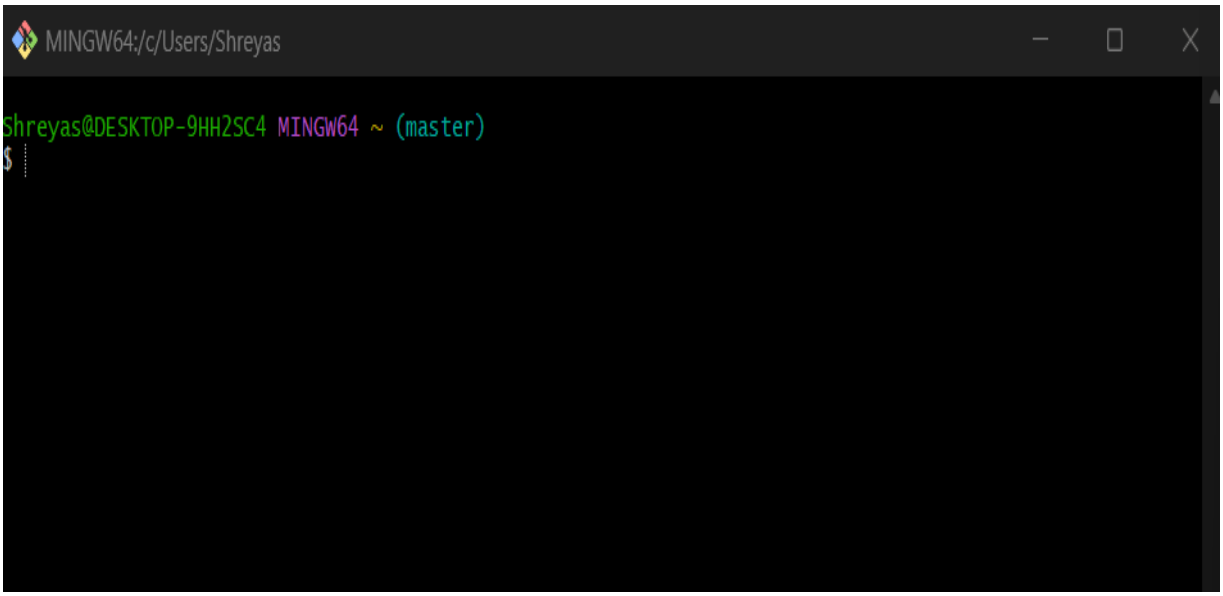


Source Code Management

LAB REPORT – 2

Step 1: Open Git Bash

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



Step 2: Check Git Version

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

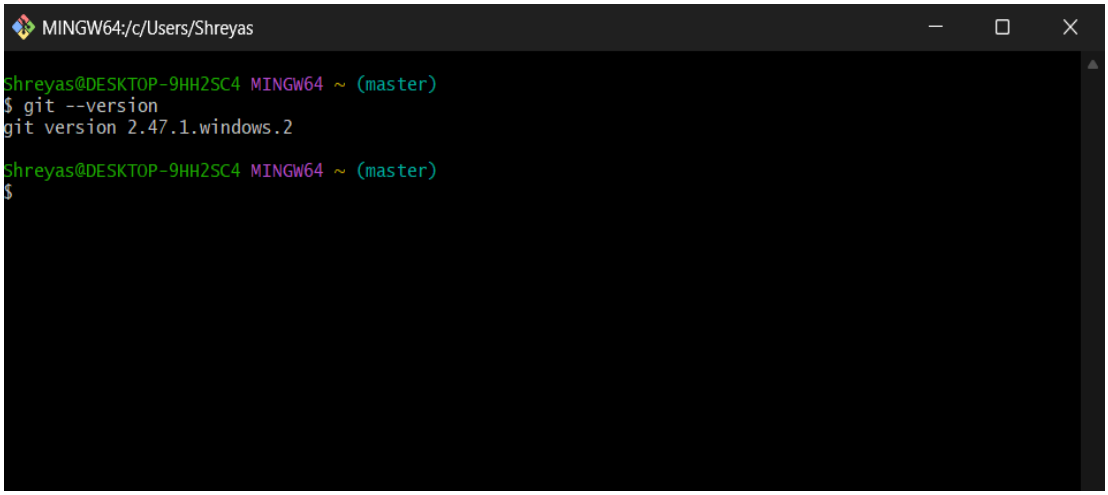
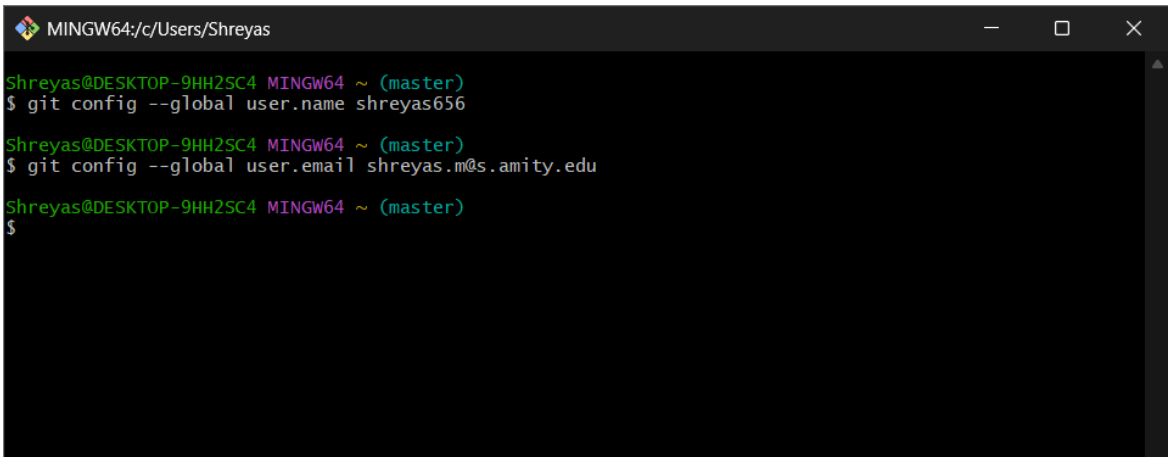


Figure – 2

Step 3: Configure Git

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

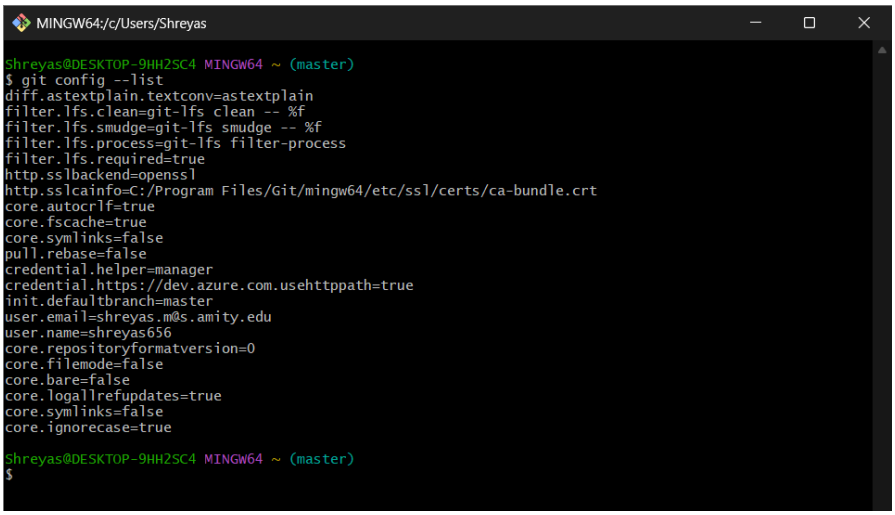
- **git config --global user.name "Your Name"**
- **git config --global user.email “your-email@example.com”**



Step 4: Verify Git Configurations

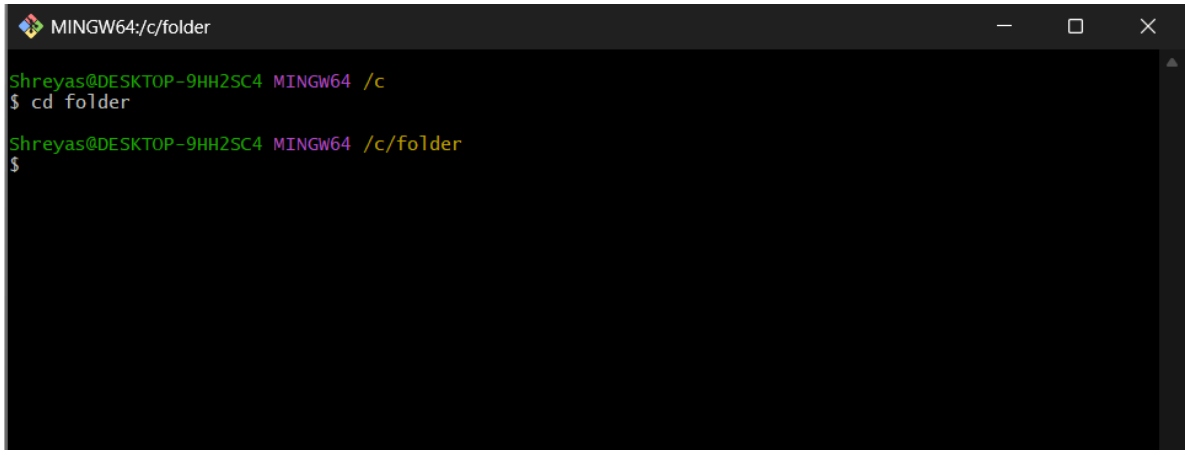
To check if the configurations were set correctly, run:

- **git config --list**



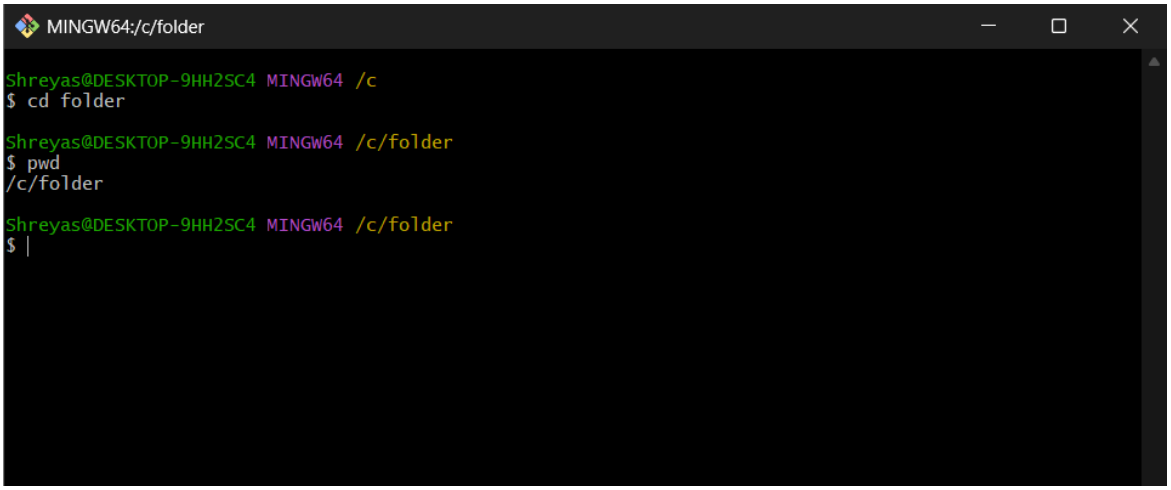
Step 5: Change Directory

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



Step 6: Print the Current Directory

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

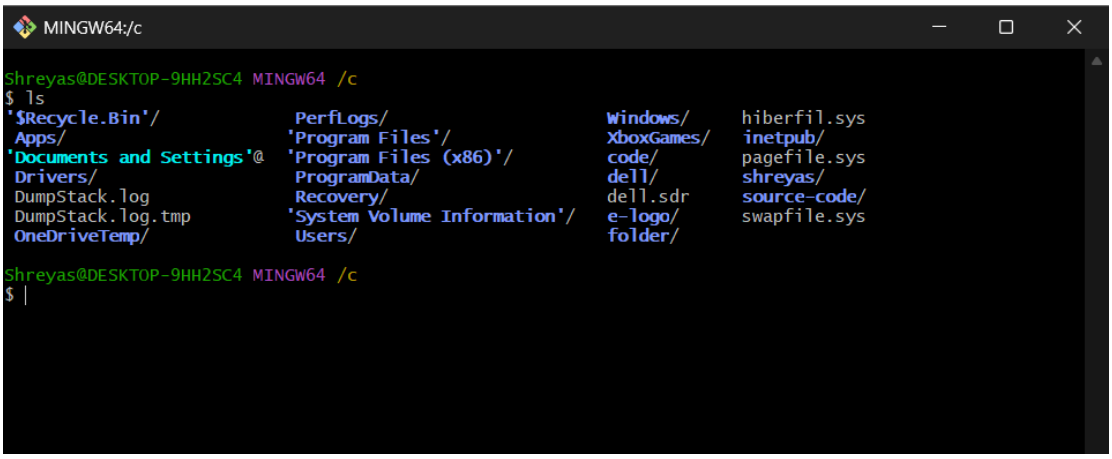


Step 7: Create a New Folder

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

Step 8: Listing the Files and Folders

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



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
$ |
```

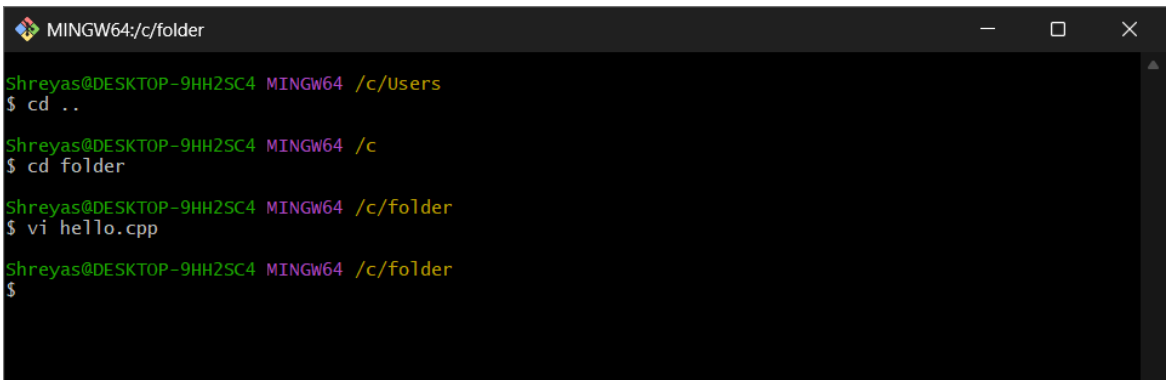
Step 10: Inside the VI Editor

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:/c/folder
#include <iostream.h>
using namespace std;
int main(){
        cout << "hello world" << endl;
        return 0;
}
~
~
~
~
~
~
```

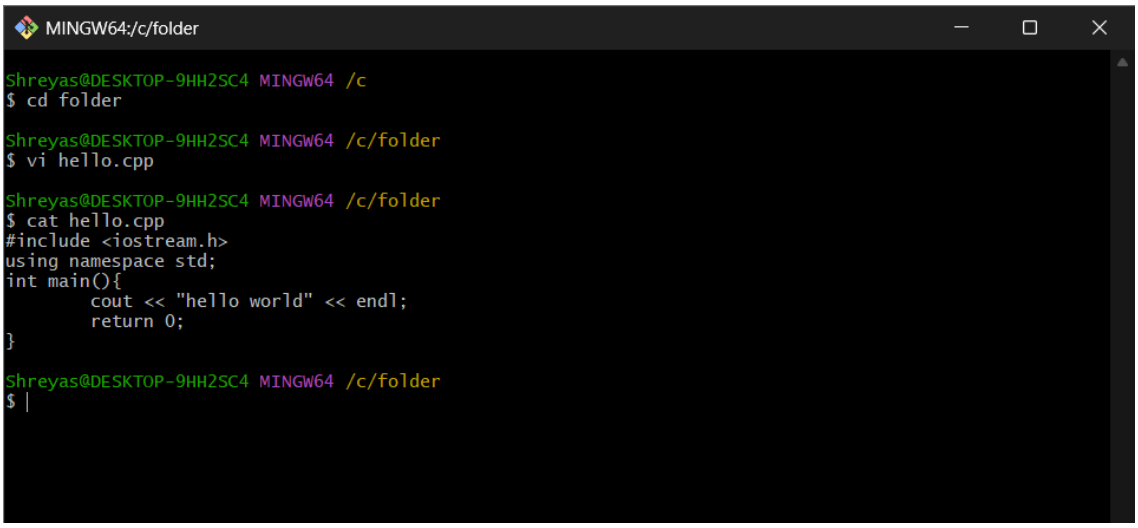
Step 11: Exiting the VI Editor

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



Step 12: Display File Contents

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



Step 13: Initialize Git in Directory

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

```
MINGW64:/c/folder

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

Shreyas@DESKTOP-9HH2SC4 MINGW64 /c/folder
$ vi hello.cpp

Shreyas@DESKTOP-9HH2SC4 MINGW64 /c/folder
$ cat hello.cpp
#include <iostream.h>
using namespace std;
int main(){
        cout << "hello world" << endl;
        return 0;
}

Shreyas@DESKTOP-9HH2SC4 MINGW64 /c/folder
$ git init
Initialized empty Git repository in C:/folder/.git/

Shreyas@DESKTOP-9HH2SC4 MINGW64 /c/folder (master)
$
```

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**.

```
MINGW64:/c/folder

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

Shreyas@DESKTOP-9HH2SC4 MINGW64 /c/folder
$ vi hello.cpp

Shreyas@DESKTOP-9HH2SC4 MINGW64 /c/folder
$ cat hello.cpp
#include <iostream.h>
using namespace std;
int main(){
        cout << "hello world" << endl;
        return 0;
}

Shreyas@DESKTOP-9HH2SC4 MINGW64 /c/folder
$ git init
Initialized empty Git repository in C:/folder/.git/

Shreyas@DESKTOP-9HH2SC4 MINGW64 /c/folder (master)
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        hello.cpp
        team_project/

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

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

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**.

```
MINGW64:/c/folder
int main(){
        cout << "hello world" << endl;
        return 0;
}

Shreyas@DESKTOP-9HH2SC4 MINGW64 /c/folder
$ git init
Initialized empty Git repository in C:/folder/.git/

Shreyas@DESKTOP-9HH2SC4 MINGW64 /c/folder (master)
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        hello.cpp
        team_project/

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

Shreyas@DESKTOP-9HH2SC4 MINGW64 /c/folder (master)
$ git add .
Shreyas@DESKTOP-9HH2SC4 MINGW64 /c/folder (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   hello.cpp
        new file:   team_project/project_1

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
  (commit or discard the untracked or modified content in submodules)
        modified:   team_project/project_1 (modified content)
```

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"**

```
Shreyas@DESKTOP-9HH2SC4 MINGW64 /c/folder (master)
$ git commit -m "initial commit"
[master (root-commit) 5cdca48] initial commit
 2 files changed, 7 insertions(+)
 create mode 100644 hello.cpp
 create mode 160000 team_project/project_1

Shreyas@DESKTOP-9HH2SC4 MINGW64 /c/folder (master)
$
```

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.

```
Shreyas@DESKTOP-9HH2SC4 MINGW64 /c/folder (master)
$ git commit -m "initial commit"
[master (root-commit) 5cdca48] initial commit
 2 files changed, 7 insertions(+)
 create mode 100644 hello.cpp
 create mode 160000 team_project/project_1

Shreyas@DESKTOP-9HH2SC4 MINGW64 /c/folder (master)
$ git log
commit 5cdca48b8867f885ff9078fd551f7f06054375ed (HEAD -> master)
Author: shreyas656 <shreyas.m@s.amity.edu>
Date:   Fri May 30 17:07:03 2025 +0530

    initial commit

Shreyas@DESKTOP-9HH2SC4 MINGW64 /c/folder (master)
$
```

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.

```
Shreyas@DESKTOP-9HH2SC4 MINGW64 /c/folder (master)
$ git log --oneline
5cdca48 (HEAD -> master) initial commit

Shreyas@DESKTOP-9HH2SC4 MINGW64 /c/folder (master)
$
```

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.

```
Shreyas@DESKTOP-9HH2SC4 MINGW64 /c/folder (master)
$ git log --oneline
5cdca48 (HEAD -> master) initial commit

Shreyas@DESKTOP-9HH2SC4 MINGW64 /c/folder (master)
$ vi hello.cpp

Shreyas@DESKTOP-9HH2SC4 MINGW64 /c/folder (master)
$ cat hello.cpp
#include <iostream.h>
using namespace std;
int main(){
        cout << "hello world" << endl;
        cout << "hello git " << endl;
        return 0;
}

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

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.

```
Shreyas@DESKTOP-9HH2SC4 MINGW64 /c/folder (master)
$ vi hello.cpp

Shreyas@DESKTOP-9HH2SC4 MINGW64 /c/folder (master)
$ cat hello.cpp
#include <iostream.h>
using namespace std;
int main(){
        cout << "hello world" << endl;
        cout << "hello git " << endl;
        return 0;
}

Shreyas@DESKTOP-9HH2SC4 MINGW64 /c/folder (master)
$ git add .
warning: in the working copy of 'hello.cpp', LF will be replaced by CRLF the next time Git touches it

Shreyas@DESKTOP-9HH2SC4 MINGW64 /c/folder (master)
$ git commit -m "second commit"
[master 877d7fb] second commit
 1 file changed, 1 insertion(+)

Shreyas@DESKTOP-9HH2SC4 MINGW64 /c/folder (master)
$
```

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.

```
Shreyas@DESKTOP-9HH2SC4 MINGW64 /c/folder (master)
$ vi hello.cpp

Shreyas@DESKTOP-9HH2SC4 MINGW64 /c/folder (master)
$ cat hello.cpp
#include <iostream.h>
using namespace std;
int main(){
        cout << "hello world" << endl;
        cout << "hello git " << endl;
        return 0;
}

Shreyas@DESKTOP-9HH2SC4 MINGW64 /c/folder (master)
$ git add .
warning: in the working copy of 'hello.cpp', LF will be replaced by CRLF the next time Git touches it

Shreyas@DESKTOP-9HH2SC4 MINGW64 /c/folder (master)
$ git commit -m "second commit"
[master 877d7fb] second commit
 1 file changed, 1 insertion(+)

Shreyas@DESKTOP-9HH2SC4 MINGW64 /c/folder (master)
$ vi hello.cpp

Shreyas@DESKTOP-9HH2SC4 MINGW64 /c/folder (master)
$ cat hello.cpp
#include <iostream.h>
using namespace std;
int main(){
        cout << "hello world" << endl;
        cout << "hello git " << endl;
        cout << "hello " << endl;
}

Shreyas@DESKTOP-9HH2SC4 MINGW64 /c/folder (master)
$ git add .
warning: in the working copy of 'hello.cpp', LF will be replaced by CRLF the next time Git touches it

Shreyas@DESKTOP-9HH2SC4 MINGW64 /c/folder (master)
$ git commit -m "third commit"
[master d2576cc] third commit
 1 file changed, 1 insertion(+), 1 deletion(-)

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

Step 6: View Git Log Again in One Line Format

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

```
Shreyas@DESKTOP-9HH2SC4 MINGW64 /c/folder (master)
$ git add .
warning: in the working copy of 'hello.cpp', LF will be replaced by CRLF the next time Git touches it

Shreyas@DESKTOP-9HH2SC4 MINGW64 /c/folder (master)
$ git commit -m "third commit"
[master d2576cc] third commit
 1 file changed, 1 insertion(+), 1 deletion(-)

Shreyas@DESKTOP-9HH2SC4 MINGW64 /c/folder (master)
$ git log --oneline
d2576cc (HEAD -> master) third commit
877d7fb second commit
5cdca48 initial commit

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

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.

```
shreyas@DESKTOP-9HH2SC4 MINGW64 /c/folder (master)
$ git diff 5cdca48 877d7fb
diff --git a/hello.cpp b/hello.cpp
index 1b66ef6..b520756 100644
--- a/hello.cpp
+++ b/hello.cpp
@@ -2,5 +2,6 @@
 using namespace std;
 int main(){
        cout << "hello world" << endl;
+       cout << "hello git " << endl;
        return 0;
 }

shreyas@DESKTOP-9HH2SC4 MINGW64 /c/folder (master)
$
```

This shows changes between the Second commit and Third commit.

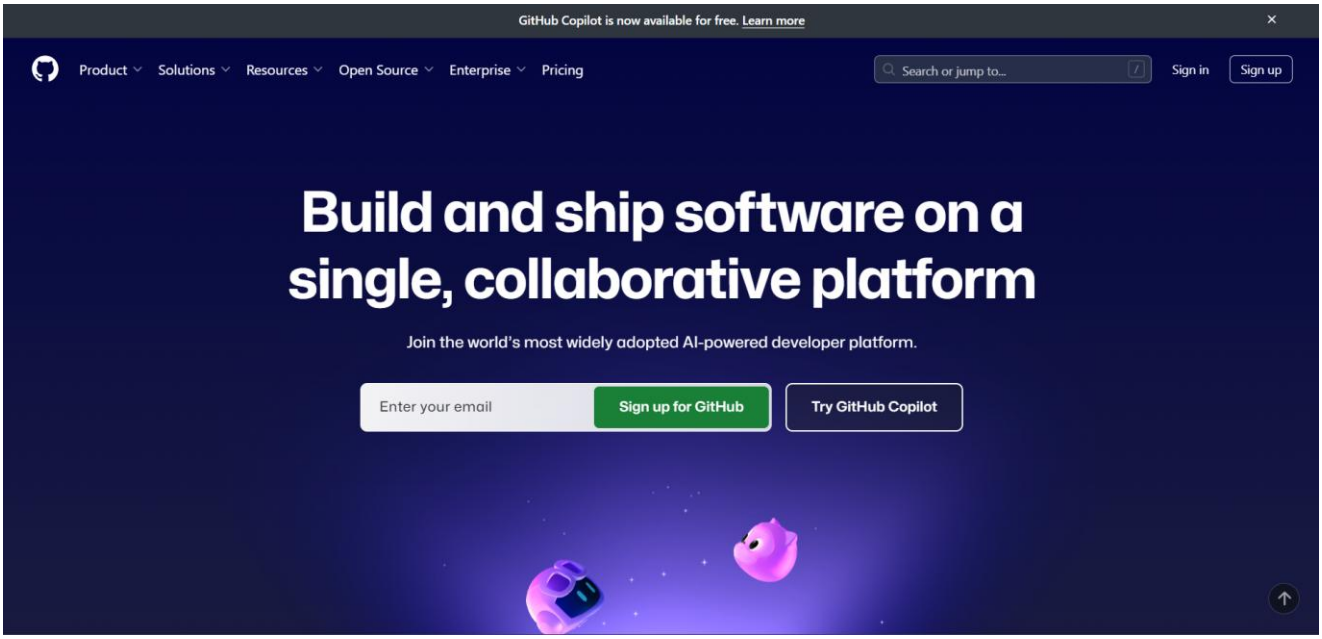
```
Shreyas@DESKTOP-9HH2SC4 MINGW64 /c/folder (master)
$ git diff 877d7fb  d2576cc
diff --git a/hello.cpp b/hello.cpp
index b520756..8c263d9 100644
--- a/hello.cpp
+++ b/hello.cpp
@@ -3,5 +3,5 @@ using namespace std;
 int main(){
        cout << "hello world" << endl;
        cout << "hello git " << endl;
-       return 0;
+       cout << "hello " << endl;
 }

Shreyas@DESKTOP-9HH2SC4 MINGW64 /c/folder (master)
$
```

Source Code Management

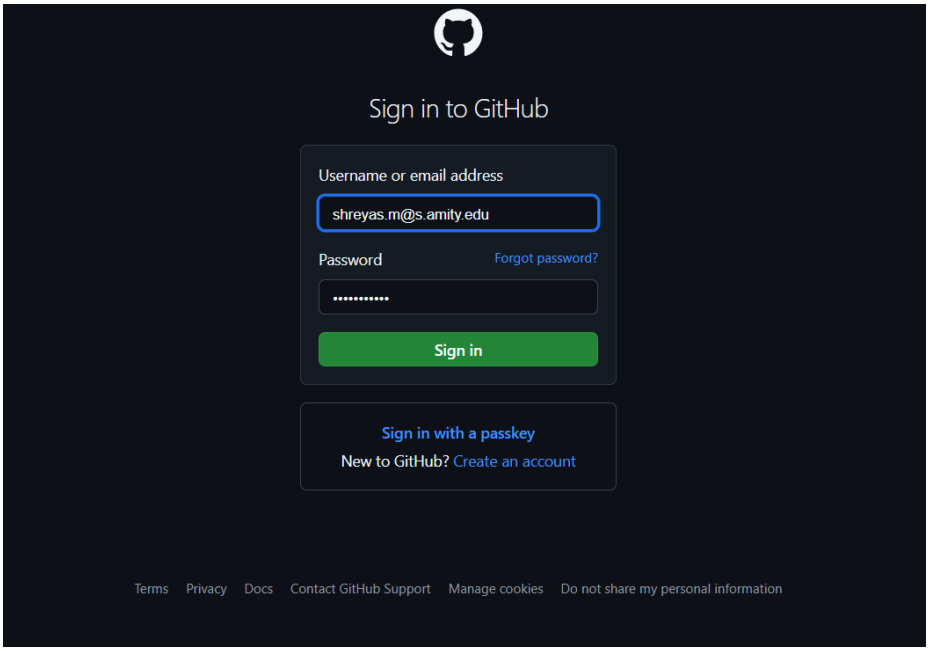
LAB REPORT – 4

Step 1: Sign in to GitHub

Open a web browser and go to github.com

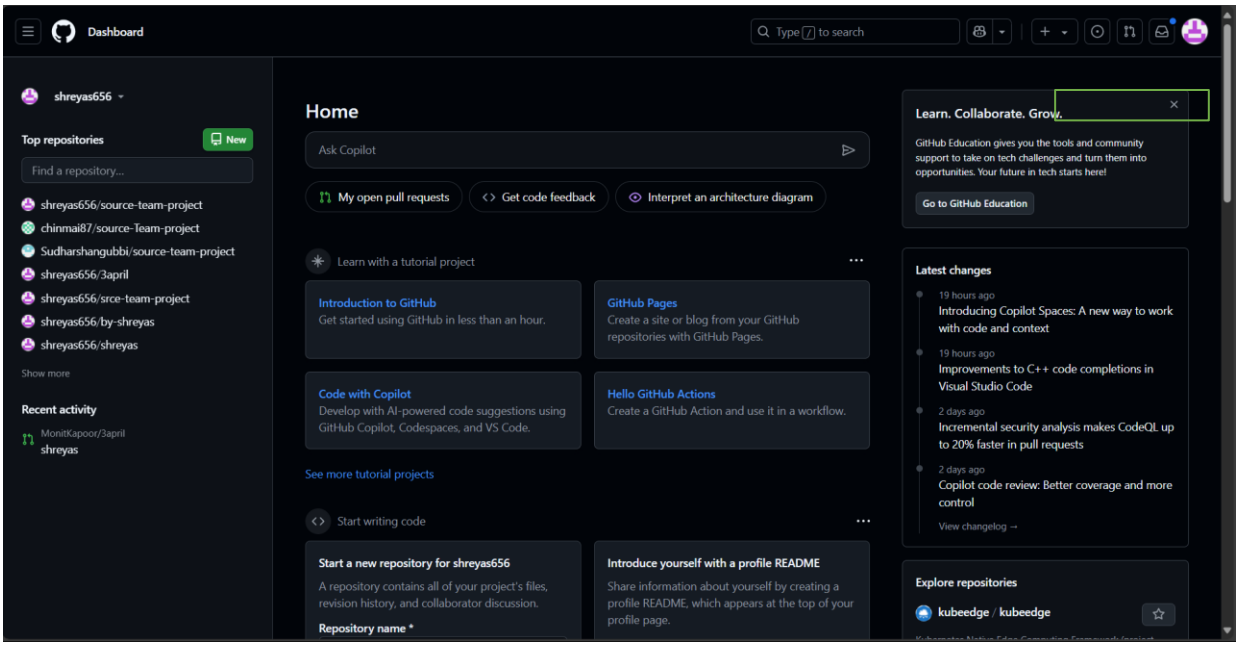


Click Sign in and enter your credentials.

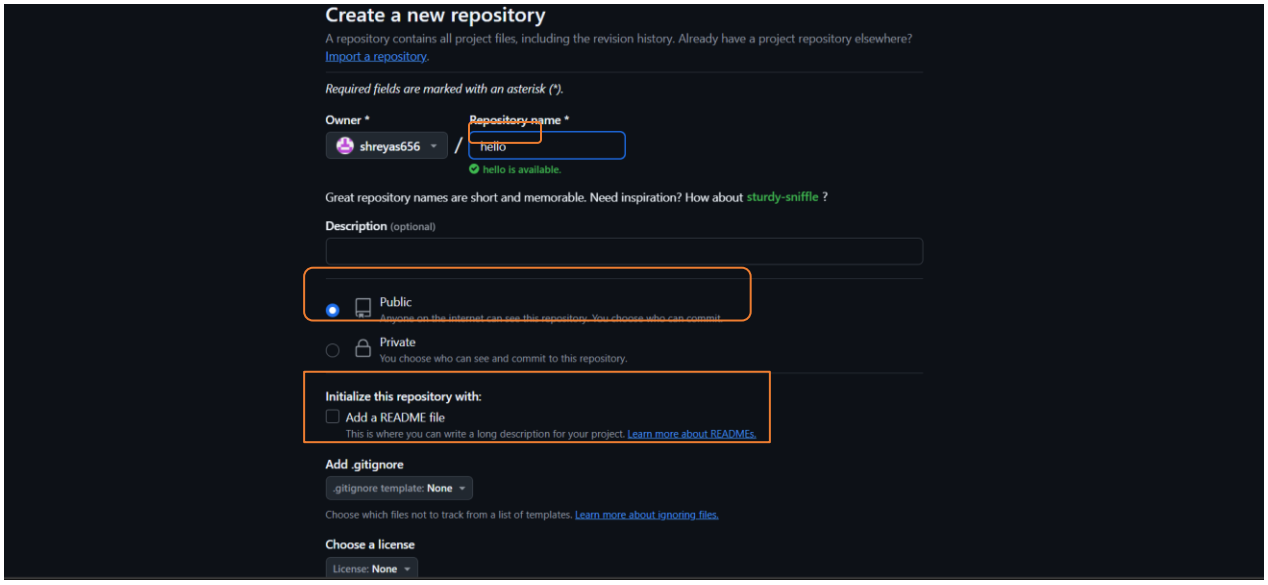


Step 2: Creating a Repository

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



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**.



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>**

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

Shreyas@DESKTOP-9HH2SC4 MINGW64 /c/folder (master)
$ git remote add origin https://github.com/shreyas656/hello
```

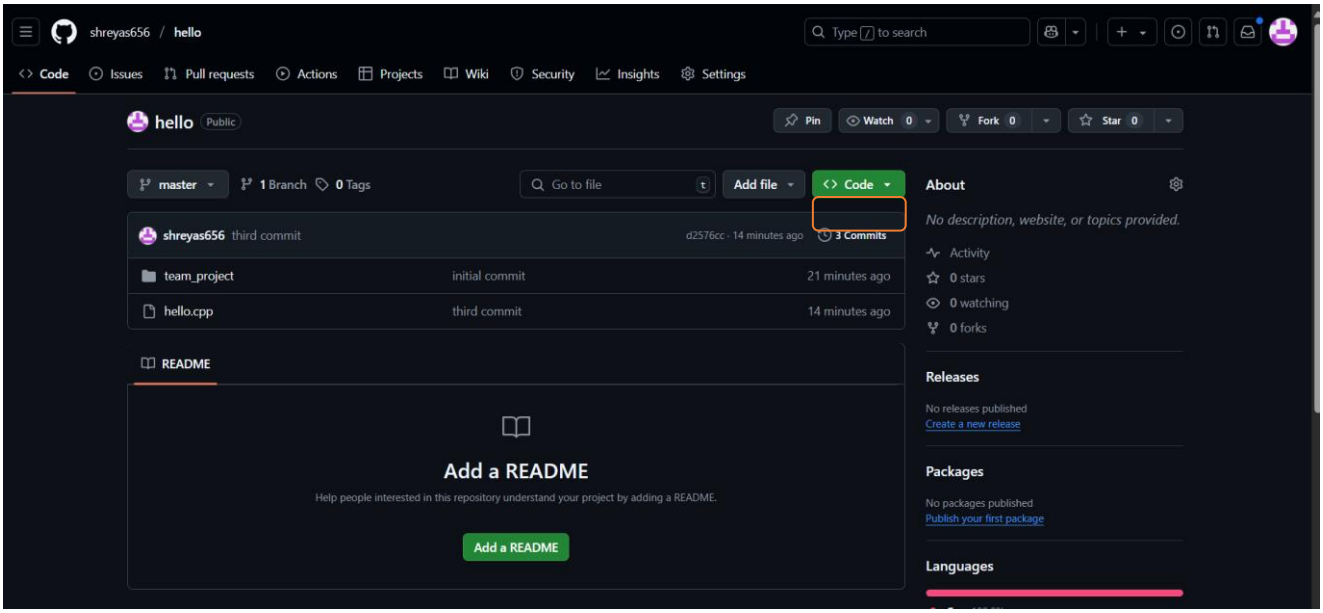
Step 4: Push Code To GitHub

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

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

Shreyas@DESKTOP-9HH2SC4 MINGW64 /c/folder (master)
$ git remote add origin https://github.com/shreyas656/hello

Shreyas@DESKTOP-9HH2SC4 MINGW64 /c/folder (master)
$ git push -u origin master
Enumerating objects: 10, done.
Counting objects: 100% (10/10), done.
Delta compression using up to 12 threads
Compressing objects: 100% (9/9), done.
Writing objects: 100% (10/10), 921 bytes | 921.00 KiB/s, done.
Total 10 (delta 2), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (2/2), done.
To https://github.com/shreyas656/hello
 * [new branch]      master -> master
branch 'master' set up to track 'origin/master'.

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

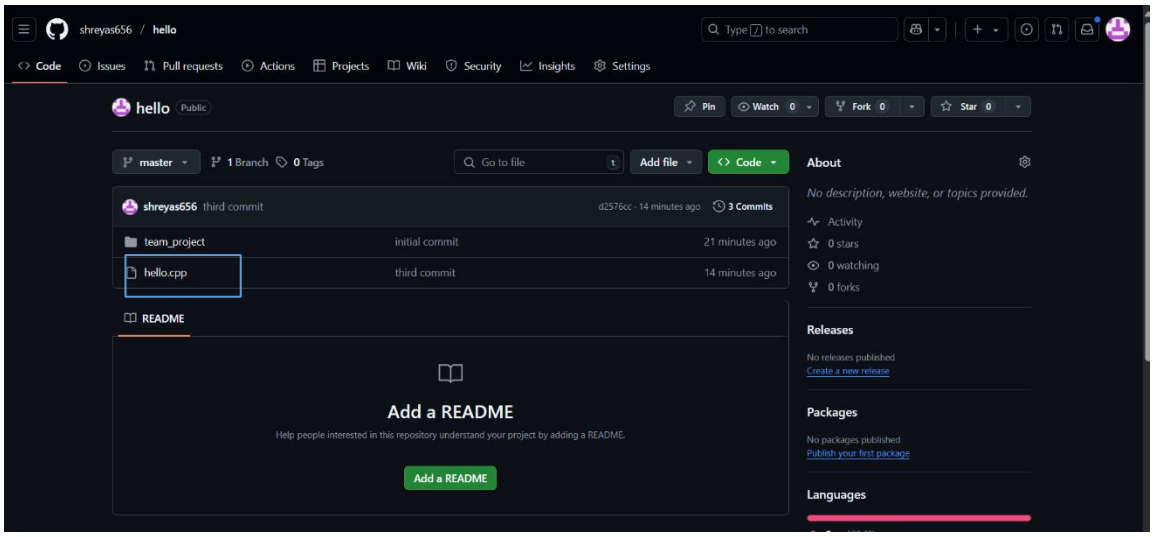
Step 5: Verify Changes on GitHub

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



Step 6: Edit the File Directly on GitHub

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



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.

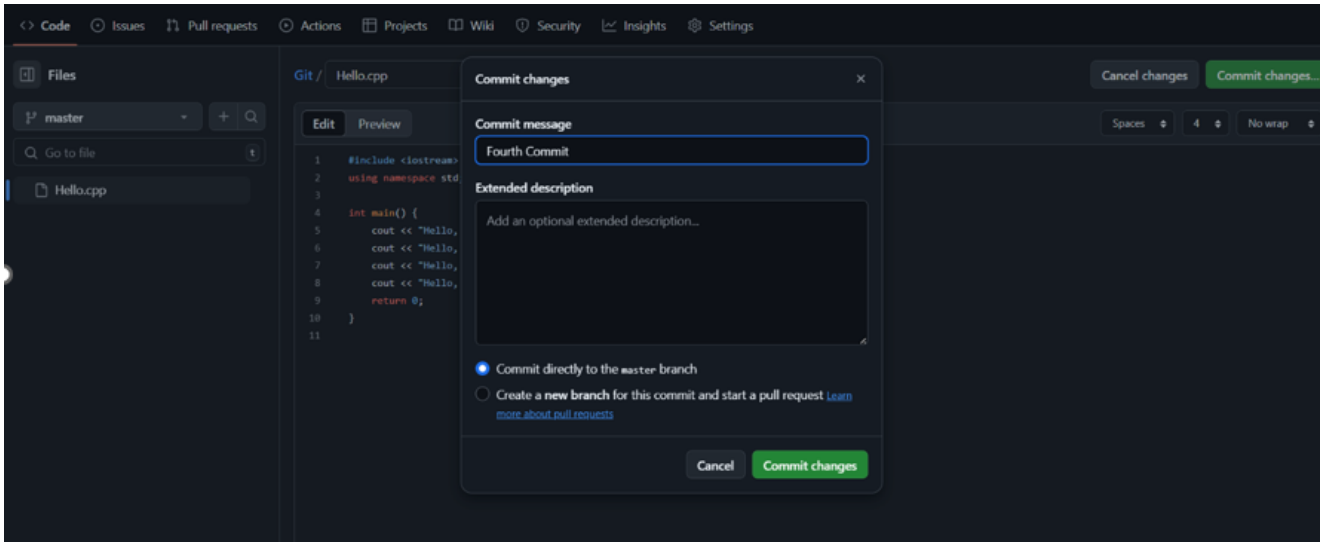


Figure – 10

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.

```
Shreyas@DESKTOP-9HH2SC4 MINGW64 /c/folder (master)
$ git pull
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
Unpacking objects: 100% (3/3), 968 bytes | 161.00 KiB/s, done.
From https://github.com/shreyas656/hello
   d2576cc..7521200  master     -> origin/master
Updating d2576cc..7521200
Fast-forward
 hello.cpp | 1 +
 1 file changed, 1 insertion(+)

Shreyas@DESKTOP-9HH2SC4 MINGW64 /c/folder (master)
$
```

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

```
Shreyas@DESKTOP-9HH2SC4 MINGW64 /c/folder (master)
$ git log
commit 75212000c815cf726ef3b9e649bc7f6bf74a27c2 (HEAD -> master, origin/master)
Author: Shreyas M <shreyas.m@s.amity.edu>
Date:   Fri May 30 17:35:43 2025 +0530

    fourth commit

commit d2576cc5d880340875d179f2635c2bc0dc83e0f3
Author: shreyas656 <shreyas.m@s.amity.edu>
Date:   Fri May 30 17:14:26 2025 +0530

    third commit

commit 877d7fb95cfd47a7a5907c8177b5813a2567798f
Author: shreyas656 <shreyas.m@s.amity.edu>
Date:   Fri May 30 17:12:30 2025 +0530

    second commit

commit 5cdca48b8867f885ff9078fd551f7f06054375ed
Author: shreyas656 <shreyas.m@s.amity.edu>
Date:   Fri May 30 17:07:03 2025 +0530

    initial commit

Shreyas@DESKTOP-9HH2SC4 MINGW64 /c/folder (master)
$
```

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
```

```
Shreyas@DESKTOP-9HH2SC4 MINGW64 /c/folder (master)
$ git checkout -b shreyas
Switched to a new branch 'shreyas'

Shreyas@DESKTOP-9HH2SC4 MINGW64 /c/folder (shreyas)
$ vi hello.cpp

Shreyas@DESKTOP-9HH2SC4 MINGW64 /c/folder (shreyas)
$ git status
On branch shreyas
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
  (commit or discard the untracked or modified content in submodules)
        modified:   hello.cpp
        modified:   team_project/project_1 (modified content)

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

Shreyas@DESKTOP-9HH2SC4 MINGW64 /c/folder (shreyas)
$
```

Step 2: Make Changes in the dev Branch

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

```
Shreyas@DESKTOP-9HH2SC4 MINGW64 /c/folder (master)
$ git checkout -b shreyas
Switched to a new branch 'shreyas'

Shreyas@DESKTOP-9HH2SC4 MINGW64 /c/folder (shreyas)
$ vi hello.cpp

Shreyas@DESKTOP-9HH2SC4 MINGW64 /c/folder (shreyas)
$
```

Step 3: Stage and Commit Changes

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

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

```
Shreyas@DESKTOP-9HH2SC4 MINGW64 /c/folder (shreyas)
$ git checkout master
M        hello.cpp
M        team_project/project_1
Switched to branch 'master'
Your branch is up to date with 'origin/master'.

Shreyas@DESKTOP-9HH2SC4 MINGW64 /c/folder (master)
$ git merge shreyas
Already up to date.

Shreyas@DESKTOP-9HH2SC4 MINGW64 /c/folder (master)
$
```

Step 6: Verify the Merge

Use cat command to check is the files are merged.

cat hello.cpp

```
Shreyas@DESKTOP-9HH2SC4 MINGW64 /c/folder (master)
$ ls
git/  hello.cpp  team_project/

Shreyas@DESKTOP-9HH2SC4 MINGW64 /c/folder (master)
$ cat hello.cpp
#include <iostream.h>
using namespace std;
int main(){
        cout << "hello world" << endl;
        cout << "hello git " << endl;
        cout << "hello hub" << endl;
        cout << "hello " << endl;
        cout << "hello gitbash" << endl;
}

Shreyas@DESKTOP-9HH2SC4 MINGW64 /c/folder (master)
$
```

Step 7: 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
```

```
Shreyas@DESKTOP-9HH2SC4 MINGW64 /c/folder (master)
$ touch .gitignore

Shreyas@DESKTOP-9HH2SC4 MINGW64 /c/folder (master)
$
```

Step 8: Viewing Hidden Files and Folders

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

```
ls -a
```

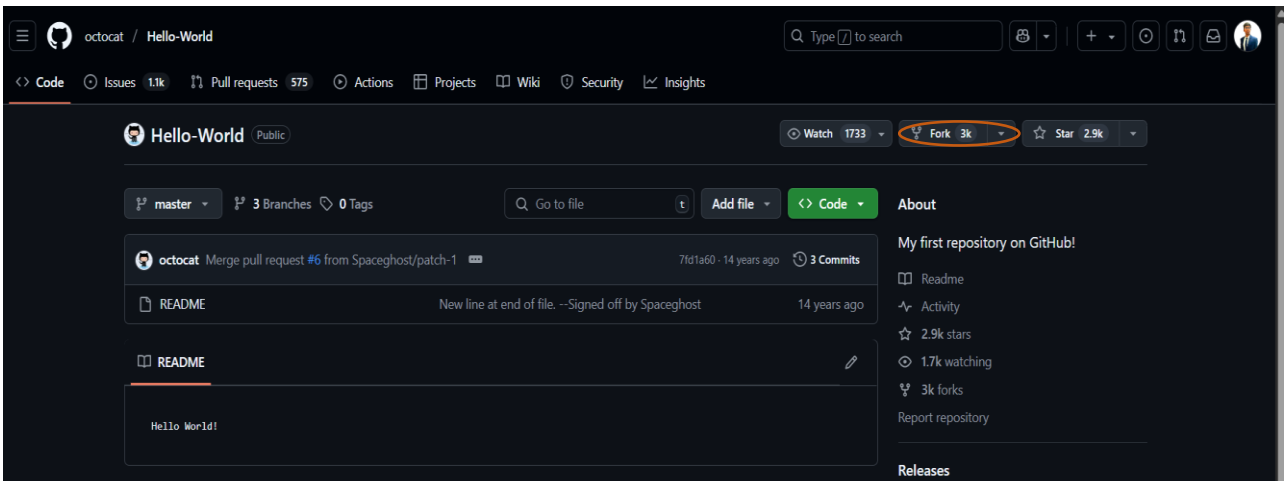
```
Shreyas@DESKTOP-9HH2SC4 MINGW64 /c/folder (master)
$ touch .gitignore

Shreyas@DESKTOP-9HH2SC4 MINGW64 /c/folder (master)
$ ls -a
./  ../  .git/  .gitignore  git/  hello.cpp  team_project/

Shreyas@DESKTOP-9HH2SC4 MINGW64 /c/folder (master)
$
```

Source Code Management

LAB REPORT – 6

Step 1: Fork a Repository on GitHub

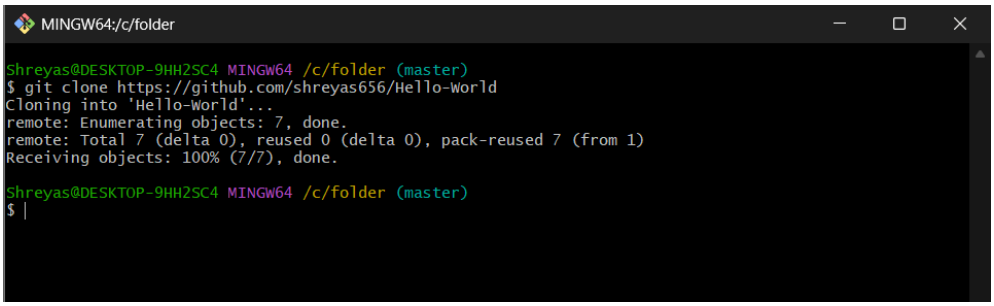
- Go to any public repository on GitHub (e.g., https://github.com/octocat/Hello-World).
- Click on the **"Fork"** button (top right corner).
- This creates a copy of the repository under **your GitHub account**.



Step 2: Clone the Forked Repository Locally

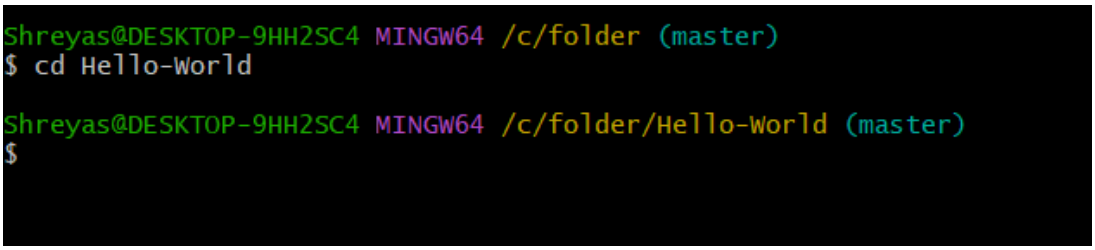
Replace your-**username** with your actual GitHub username.

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



Step 3: Change Directory to the Cloned Repo

```
cd Hello-World
```



Step 4: Add a New File or Modify Existing One

Vi hello.cpp

```
Shreyas@DESKTOP-9HH2SC4 MINGW64 /c/folder (master)
$ cd Hello-World

Shreyas@DESKTOP-9HH2SC4 MINGW64 /c/folder/Hello-World (master)
$ ls
README

Shreyas@DESKTOP-9HH2SC4 MINGW64 /c/folder/Hello-World (master)
$ vi hello.cpp

Shreyas@DESKTOP-9HH2SC4 MINGW64 /c/folder/Hello-World (master)
$ |
```

Step 5: Stage and Commit Your Changes

git add .

git commit -m "First Commit"

```
Shreyas@DESKTOP-9HH2SC4 MINGW64 /c/folder/Hello-World (master)
$ vi hello.cpp

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

Shreyas@DESKTOP-9HH2SC4 MINGW64 /c/folder/Hello-World (master)
$ git commit -m "first commit"
[master ed43303] first commit
 1 file changed, 2 insertions(+)
 create mode 100644 hello.cpp

Shreyas@DESKTOP-9HH2SC4 MINGW64 /c/folder/Hello-World (master)
$ |
```

Step 6: Push Changes to Your Forked GitHub Repo

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

git push origin master

```
Shreyas@DESKTOP-9HH2SC4 MINGW64 /c/folder/Hello-World (master)
$ git commit -m "first commit"
[master ed43303] first commit
 1 file changed, 2 insertions(+)
 create mode 100644 hello.cpp

Shreyas@DESKTOP-9HH2SC4 MINGW64 /c/folder/Hello-World (master)
$ git push origin master
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 12 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 276 bytes | 276.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/shreyas656/Hello-World
   7fd1a60..ed43303  master -> master

Shreyas@DESKTOP-9HH2SC4 MINGW64 /c/folder/Hello-World (master)
$ |
```

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.

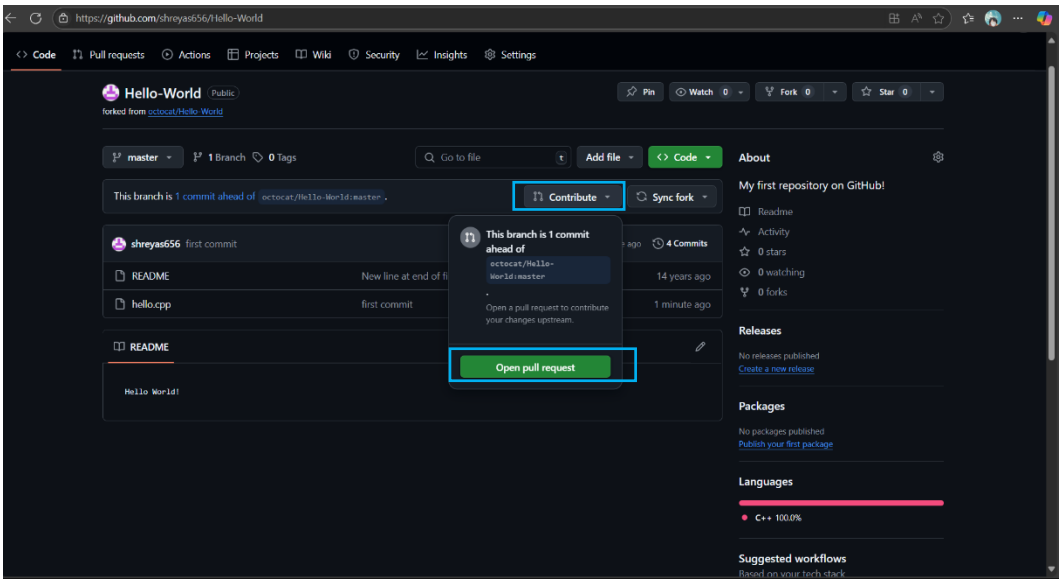


Figure - 8