**Source Code Management**

**LAB REPORT – 1**

**Overview of Git:**

Git is a [distributed version control](https://en.wikipedia.org/wiki/Distributed_version_control) [system](https://en.wikipedia.org/wiki/Software_system) that tracks versions of files. It is often used to control [source code](https://en.wikipedia.org/wiki/Source_code) by [programmers](https://en.wikipedia.org/wiki/Programmer) 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.

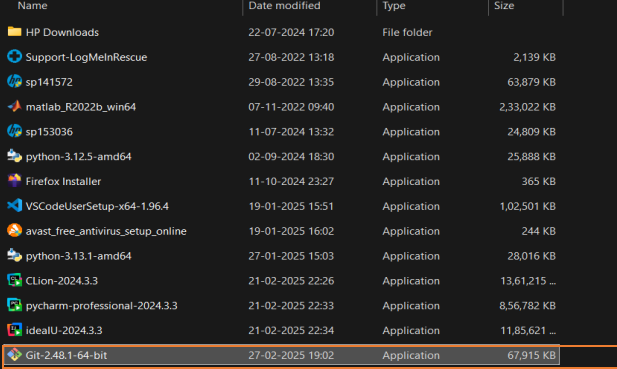
**Step 1: Downloading Git**

1. Open your web browser and navigate to the official Git website: [https://gitscm.com](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.

 **Figure - 1**

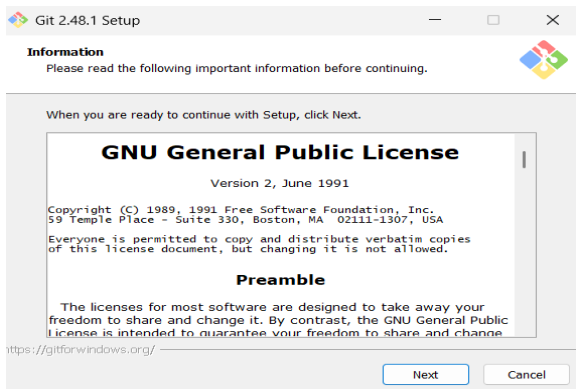
**Step 2: Running the Git Installer**

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

 **Figure – 2**

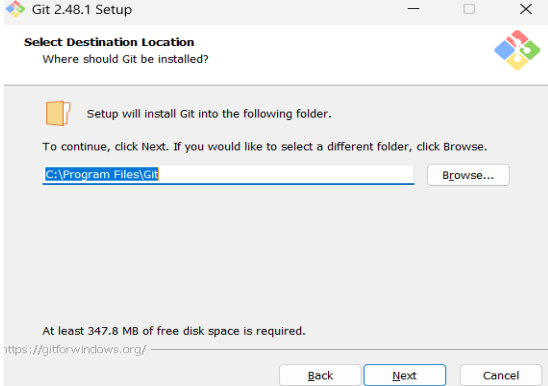
**Step 3: License (Terms and Conditions)**

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

 **Figure - 3**

**Step 4: Choose Installation Location**

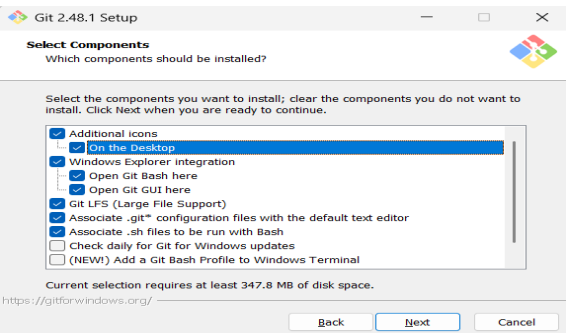
Choose the installation location (default is **C:\Program Files\Git**) and click **Next**.



**Figure – 4**

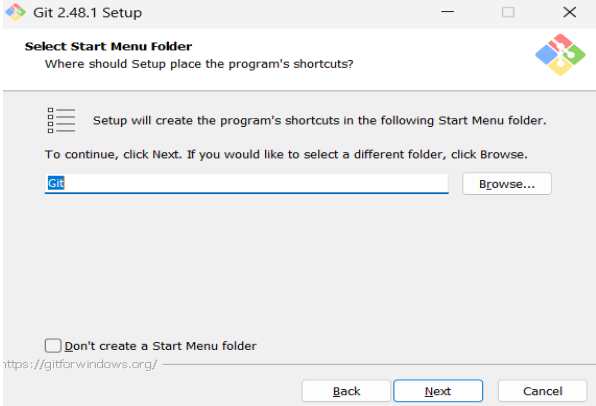
**Step 5: Select the Components**

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

**Figure – 5**

**Step 6: Select Start Menu Folder**

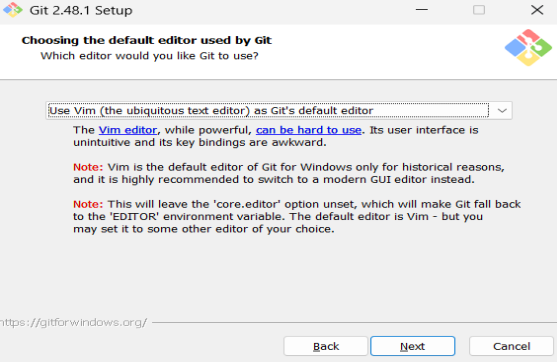
Choose the Start Menu folder where Git shortcuts will be placed. By default, the folder is named **"Git"**. Keep the default name and click **Next** to Proceed.



**Figure – 6**

**Step 7: Choose the Text Editor**

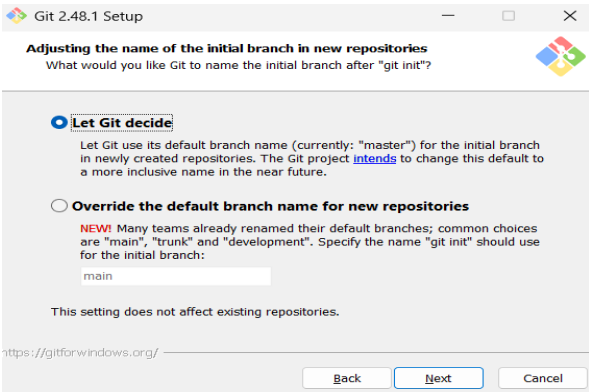
Choose a default text editor (select **Vim**) and Click **Next**.



**Figure – 7**

**Step 8: Adjusting Initial Branch Name**

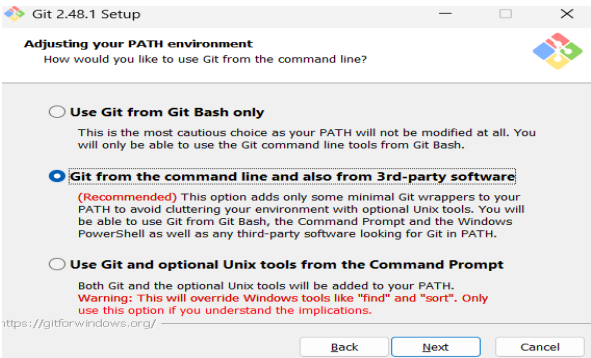
Choose the default name for the first branch when initializing a new Git repository. Go with **‘Let Git Decide’** option setting the branch as **Master** branch and proceed with **Next**.



**Figure – 8**

**Step 9: Adjusting PATH Environment**

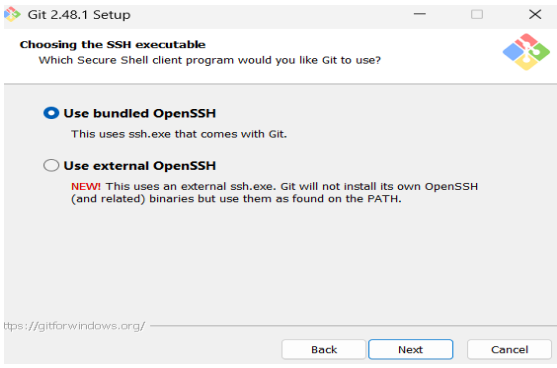
Select **Git from the command line and also from third-party software** (recommended). Click **Next**.



**Figure – 9**

**Step 10: Choosing the SSH Executable**

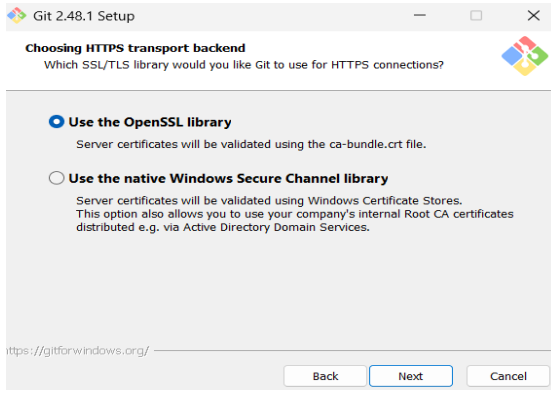
Select "**Use bundled OpenSSH**" for better compatibility and Click on **Next**.



**Figure - 10**

**Step 11: Choosing the HTTP Transport Background**

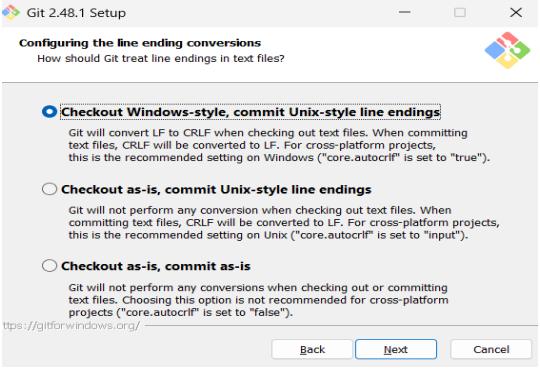
Choose Use the **OpenSSL library** (default) and Click **Next**.



**Figure – 11**

**Step 12: Configuring Line Ending Configs**

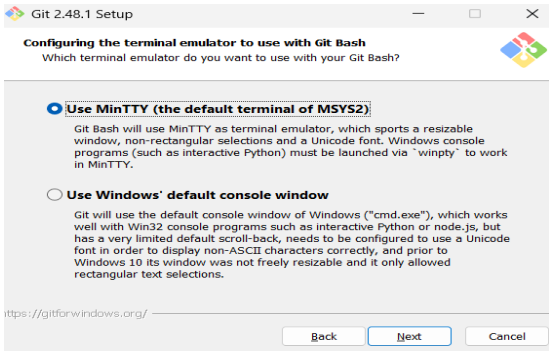
Select **Checkout Windows-style, commit Unix-style line endings** (recommended) and Click **Next**.



**Figure - 12**

**Step 13: Configuring the Terminal Emulator**

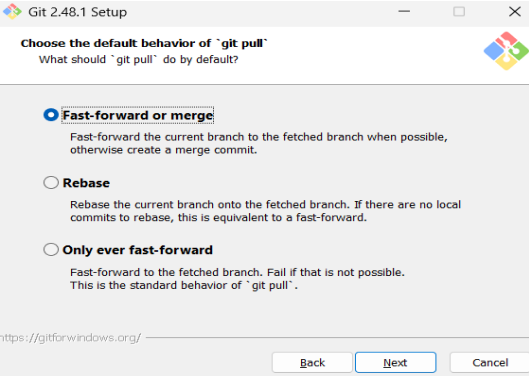
Select **Use MinTTY** **(default terminal for MSYS2**) and Click **Next**.



**Figure- 13**

**Step 14: Choosing the Default Behaviour**

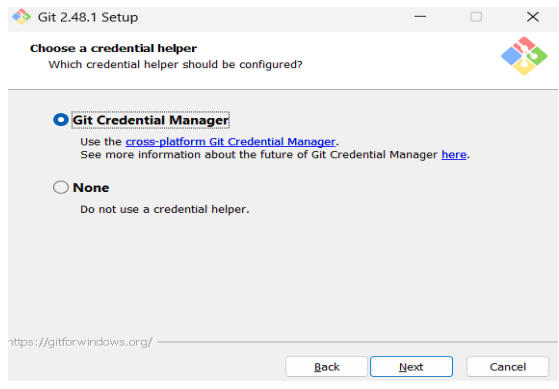
Select **Fast-forward or Merge** (recommended) option and click **Next**.



**Figure – 14**

**Step 15: Choosing a Credential Helper**

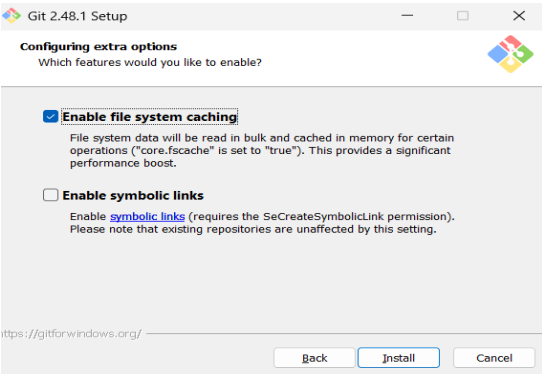
Select **Git Credential Manager** (recommended) and Click **Next**.



**Figure – 15**

**Step 16: Configuring Extra Options**

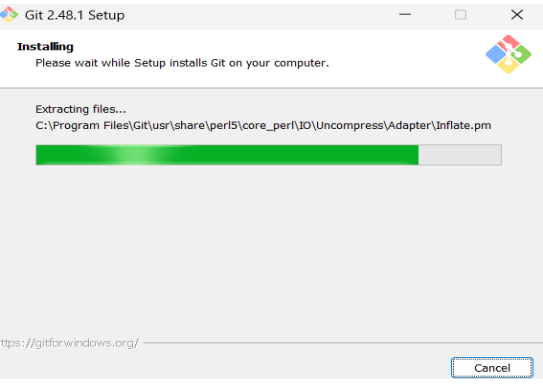
Select **Enable file system caching** (recommended) and Click on **Install**.



**Figure – 16**

**Step 17: Installation Overview**

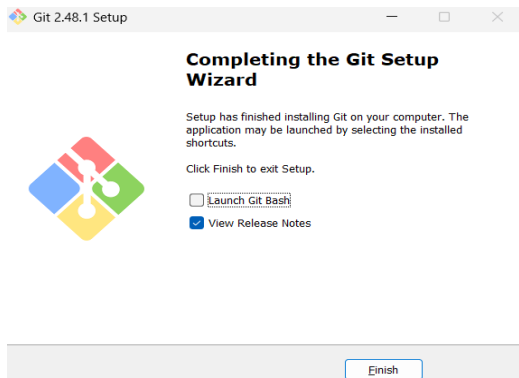
A progress bar (**green bar**) will appear, indicating that Git is being installed. Wait for the installation to complete. This may take a few minutes.



**Figure – 17**

**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**.



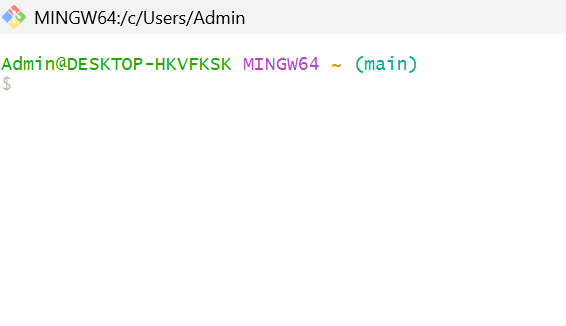
**Figure – 18**

**Source Code Management**

**LAB REPORT – 2**

**Step 1: Open Git Bash**

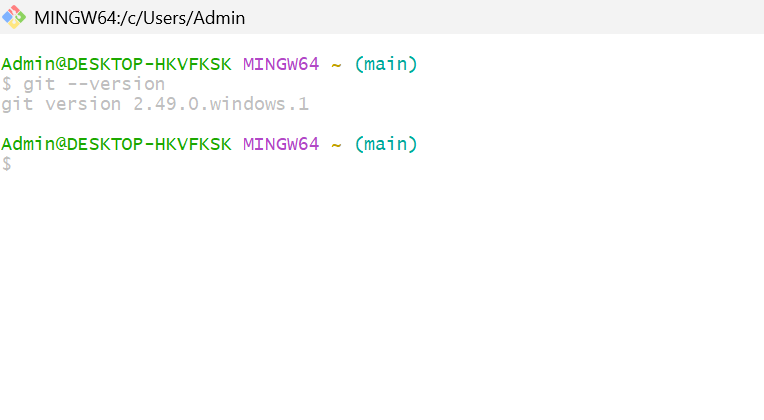
Open **Git Bash** from the Start menu or by searching for it.



**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**](mailto: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**

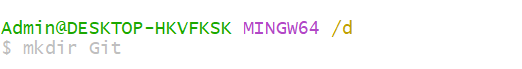
To print the full path of your current Directory use the ‘**pwd’** command.



**Figure – 6**

**Step 7: Create a New Folder**

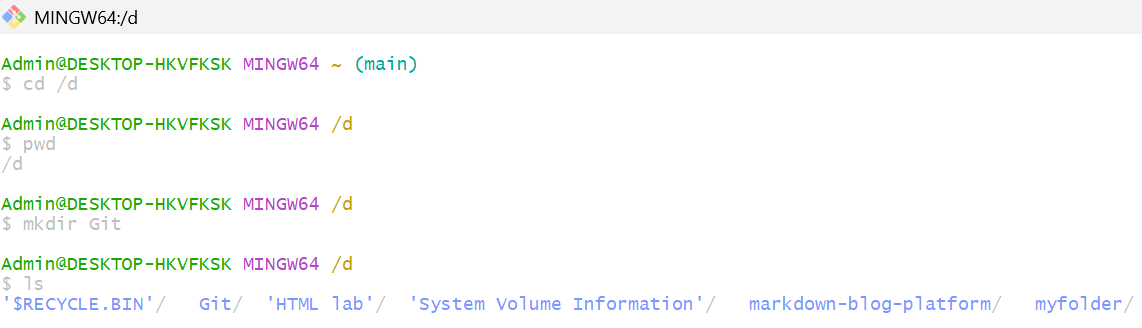
To Create a new folder in the Directory, use the command: **mkdir** folder-name.



**Figure – 7**

**Step 8: Listing the Files and Folders**

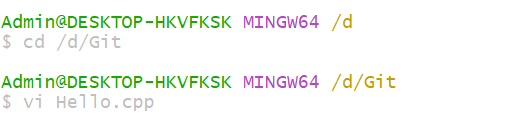
To Display the list of all files and folders in the current directory use the **‘ls’** command.



**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.



**Figure – 9**

**Step 10: Inside the VI Editor**

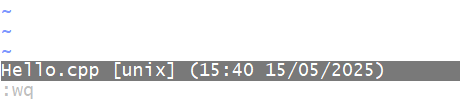
Once typed Git opens the **‘vi’** editor to create or edit a file named **Hello.cpp.** Press **i** to enter **INSERT** mode. Now start typing your code in the **vi** Editor.



**Figure – 10**

**Step 11: Exiting the VI Editor**

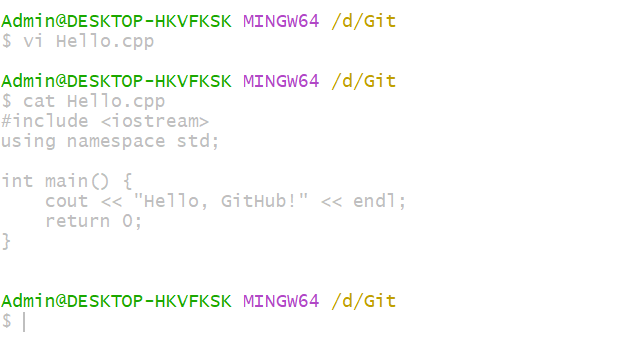
Once done with the code Press **ESC** to exit **INSERT** mode and type **:wq** and press **Enter** to save and exit.



**Figure – 11**

**Step 12: Display File Contents**

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



**Figure – 12**

**Step 13: Initialize Git in Directory**

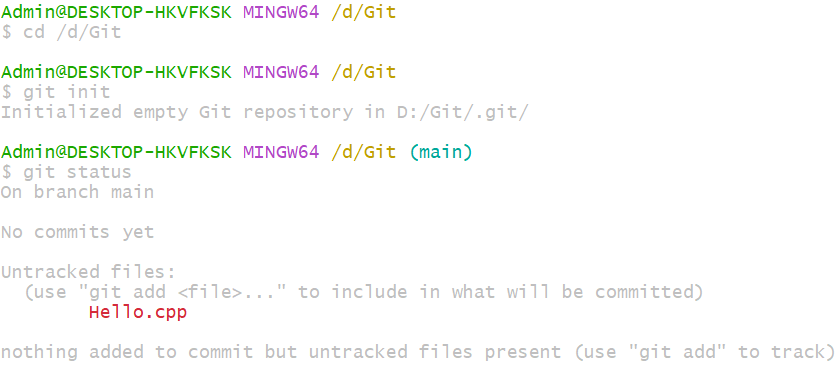
To turn the directory into a Git repository, run: **git init**



**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**

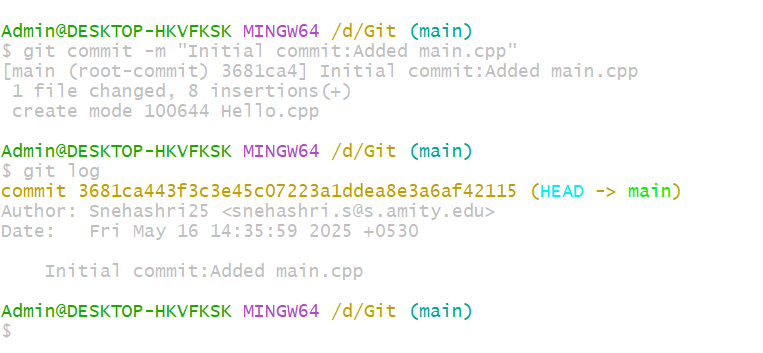
**LAB REPORT – 3**

**Source Code Management**

**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).

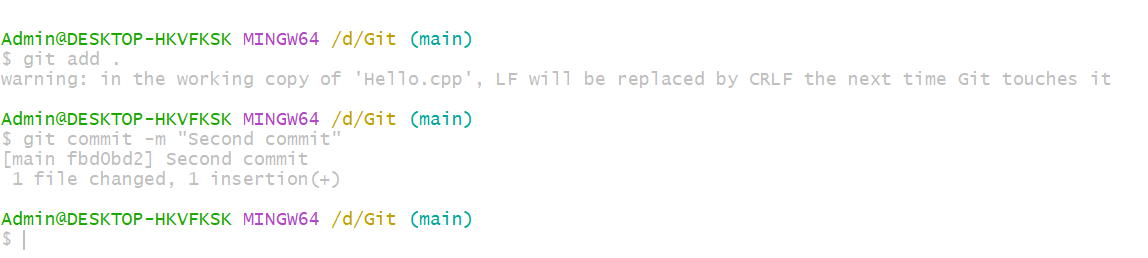
· Save the file and display it using the **cat** command.



**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.



**Figure – 5**

**Step 6: View Git Log Again in One Line Format**

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



**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.



**Figure – 7**

This shows changes between the Second commit and Third commit.



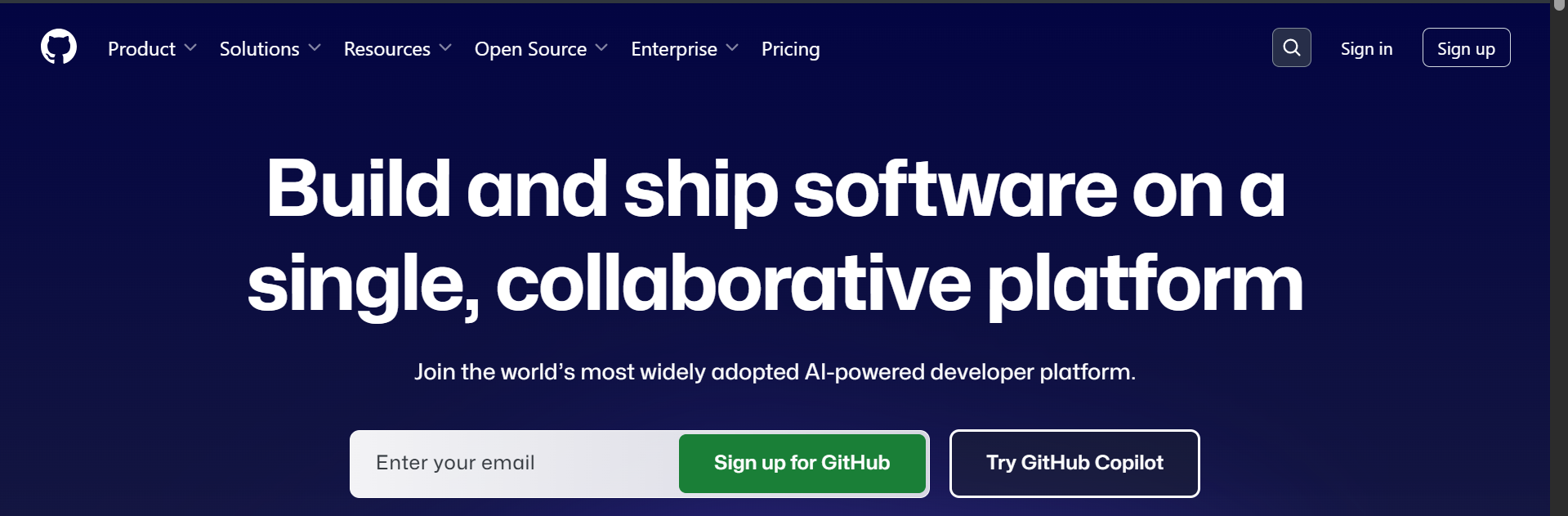
**Figure – 8**

**Source Code Management**

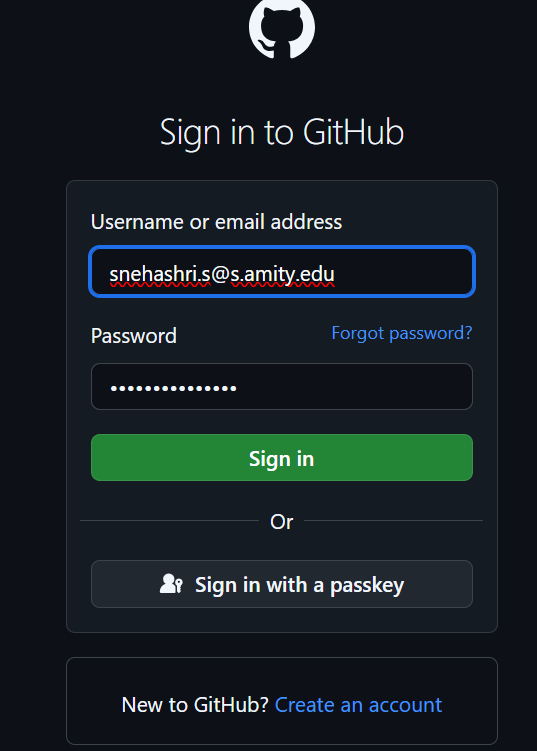
**LAB REPORT – 4**

**Step 1: Sign in to GitHub**

Open a web browser and go to [github.com](https://github.com/)

**Figure – 1**

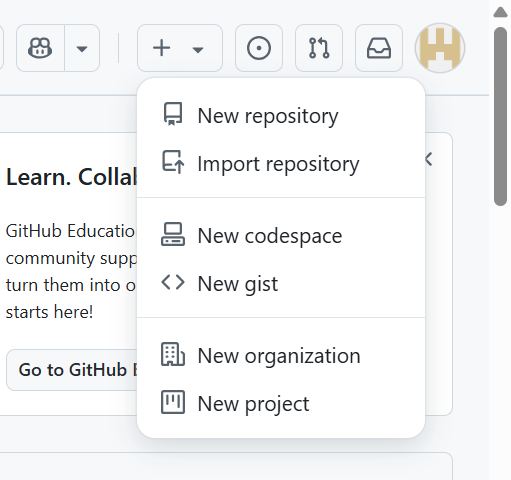
Click Sign in and enter your credentials.



**Figure – 2**

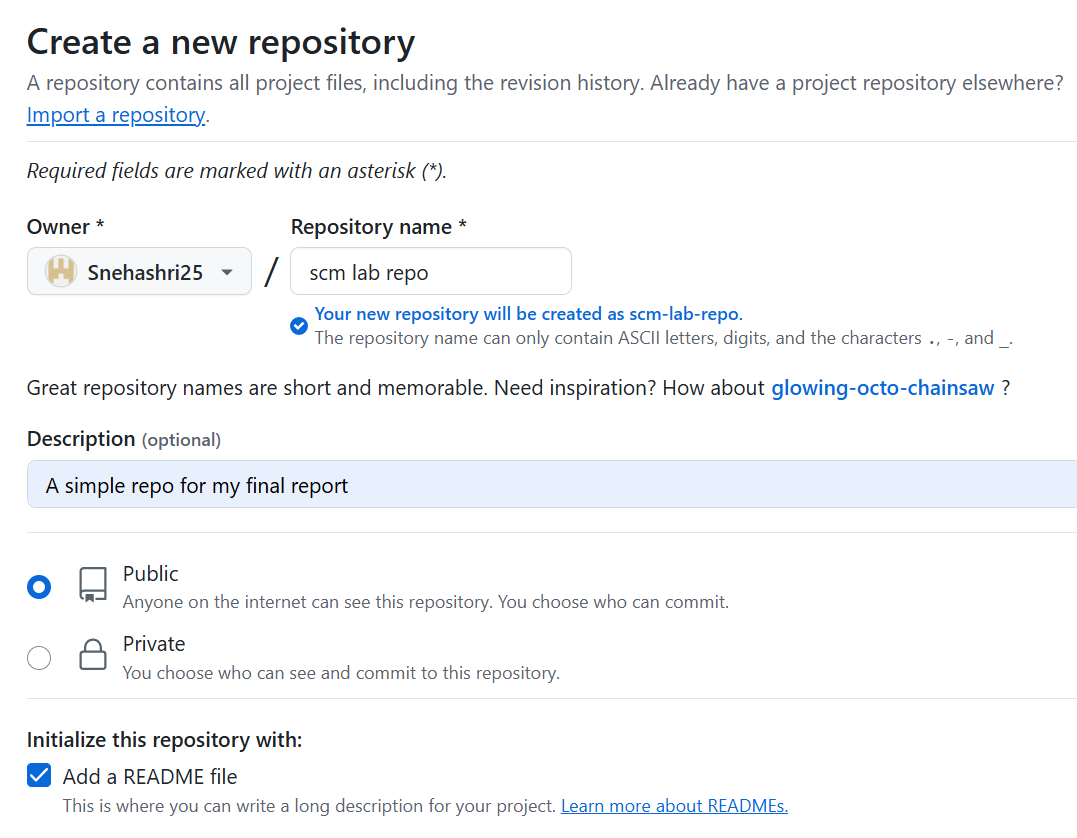
**Step 2: Creating a Repository**

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**.



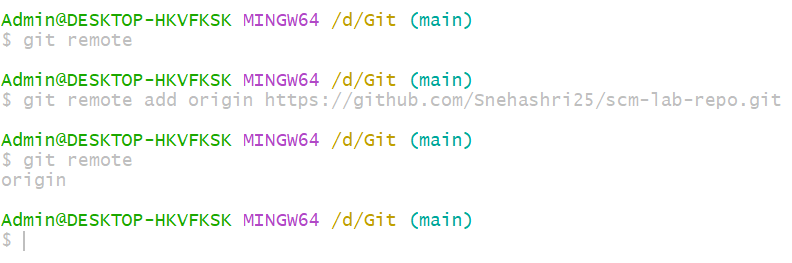
**Figure – 4**

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

Add the GitHub repository as a remote:

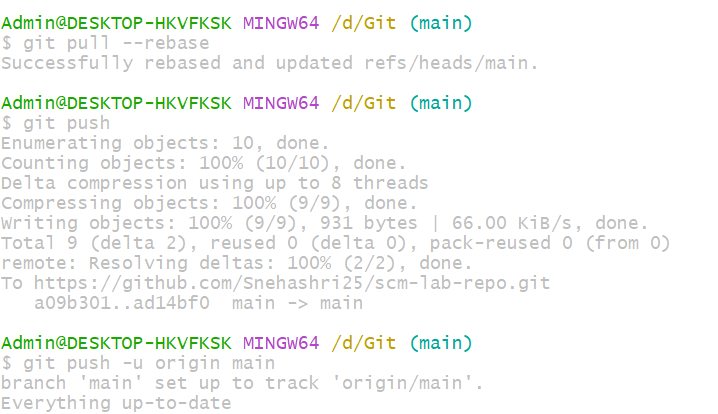
* **git remote**
* **git remote add origin <repository-URL>**



**Figure – 5**

**Step 4: Push Code To GitHub**

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



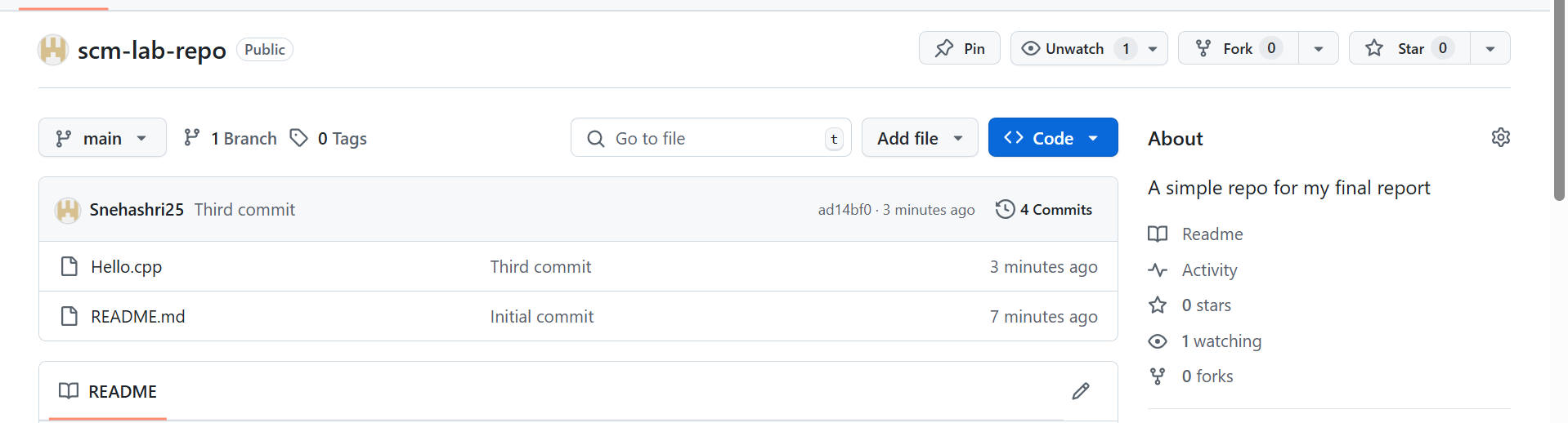
**Figure – 6**

**Step 5: Verify Changes on GitHub**

1. Open **GitHub** in your browser.

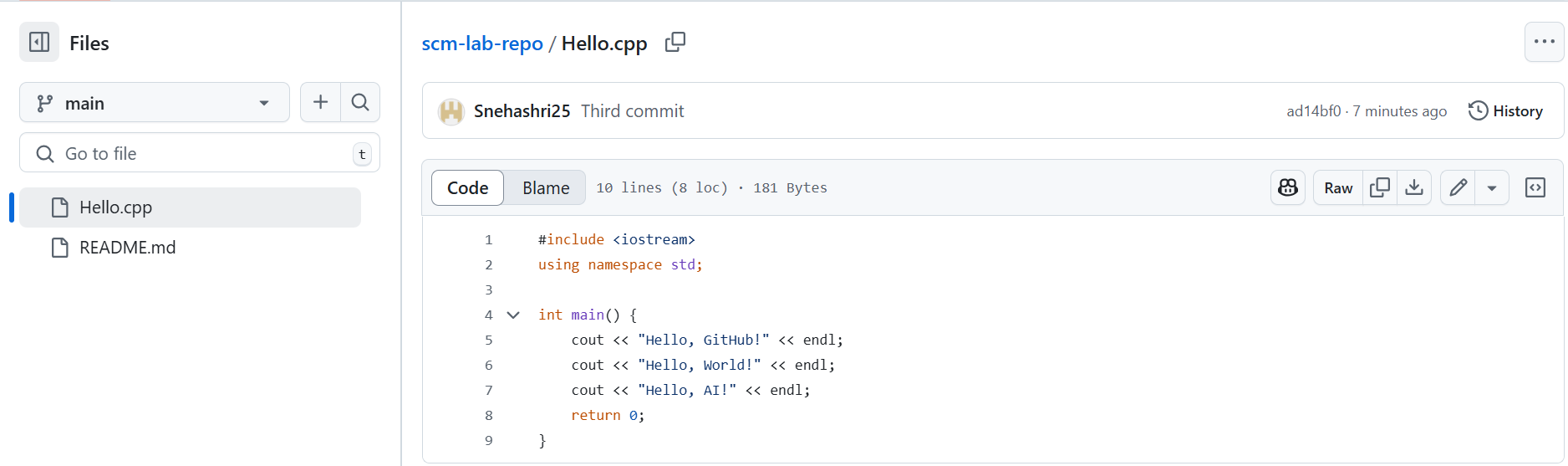
2. Go to your repository.

3. Refresh the page – your files should be visible in the repository.

**Figure – 7**

**Step 6: Edit the File Directly on GitHub**

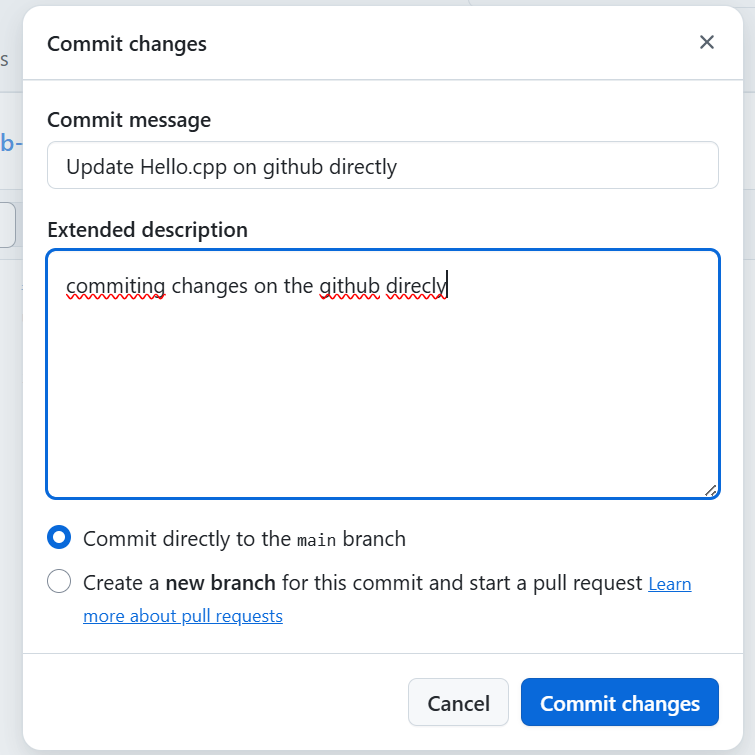
1. Click on Hello.cpp file in your GitHub repository.

 **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.



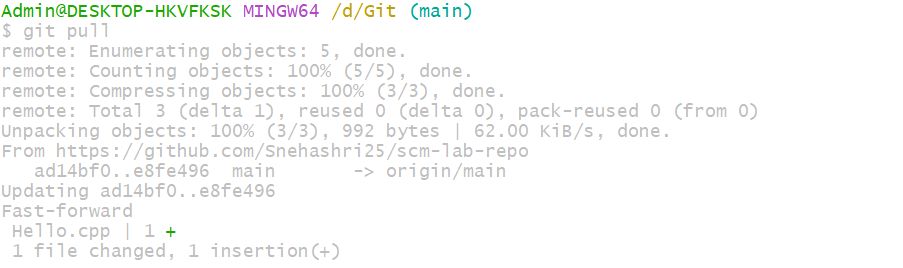
**Figure – 10**

**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.



**Figure – 11**

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



**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**



**Figure - 1**

**Step 2: Make Changes in the dev Branch**

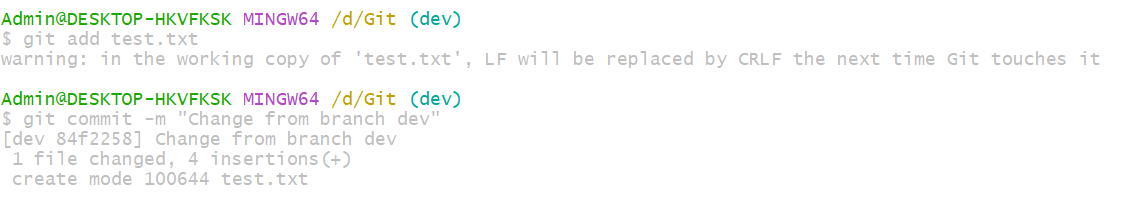
Open the **test.txt** file and make some changes.



**Figure – 2**

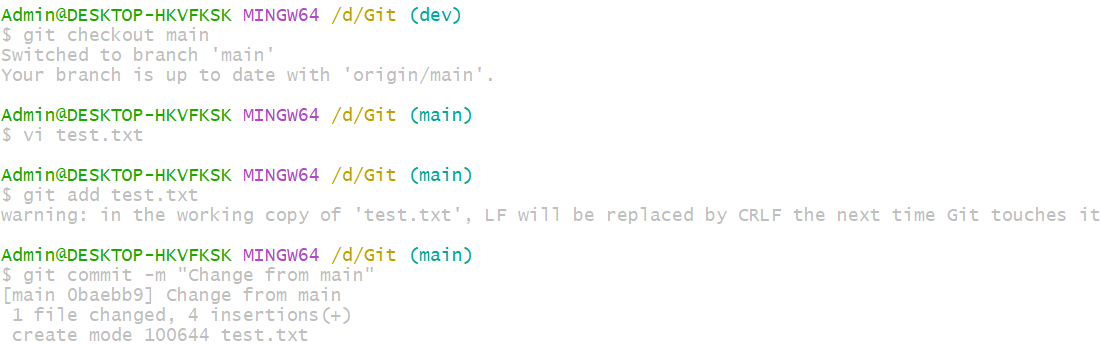
**Step 3: Stage and Commit Changes**

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

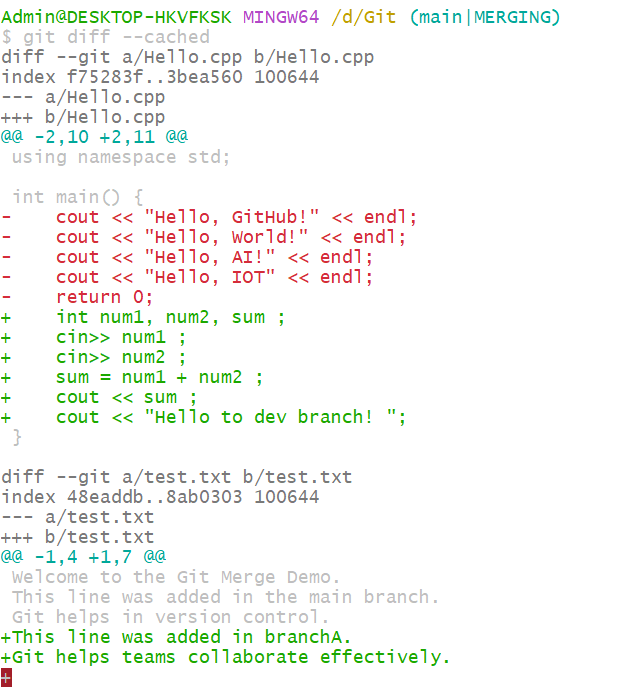
**Figure – 3**

**Step 4: Switch Back to master Branch**

**git checkout main**

**Figure – 4**

**Step 5 : git diff :** Reviewing Changes Before Finalizing Merge

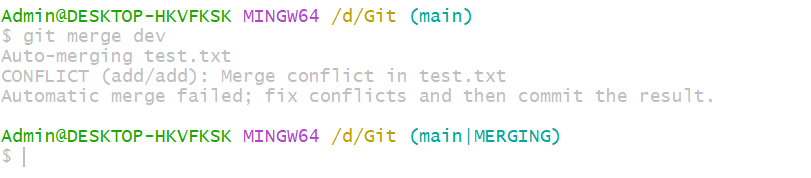


**Figure – 5**

**Step 6: Merge dev into master**

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

**git merge dev**

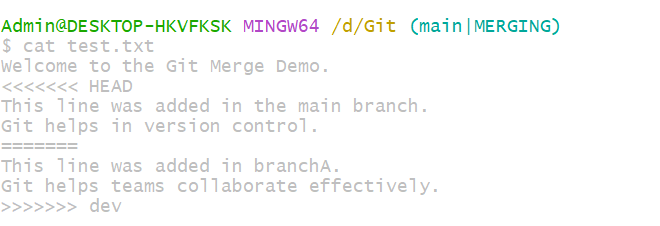


**Figure - 6**

**Step 7: Verify the Merge**

Use cat command to check is the files are merged.

**cat test.txt**



**Figure – 7**

**Step 8: Run the Git Merge Tool**

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

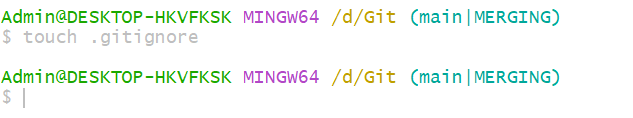
**git mergetool**

 **Figure – 8**

**Step 9: 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.

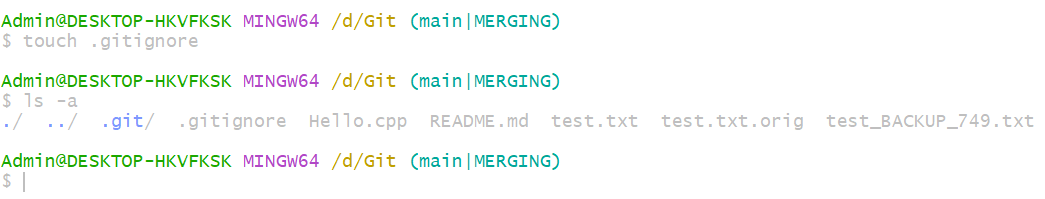
**touch .gitignore**



**Figure – 9**

**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**

**Figure – 8**

**Source Code Management**

**LAB REPORT – 6**

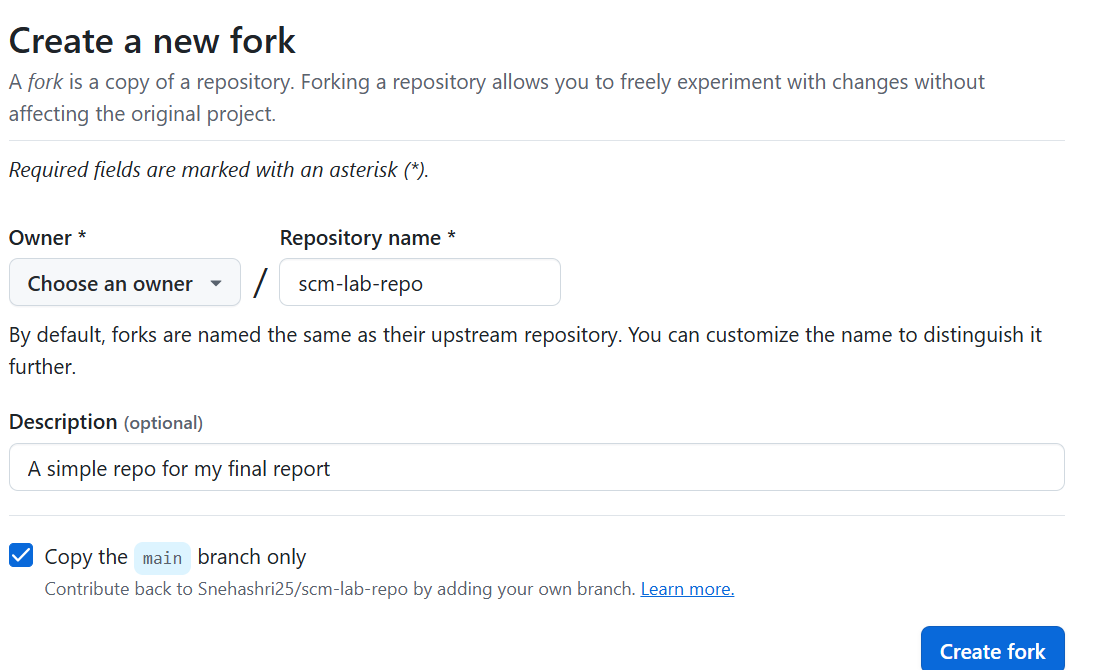
**Step 1: Fork a Repository on GitHub**

· 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**

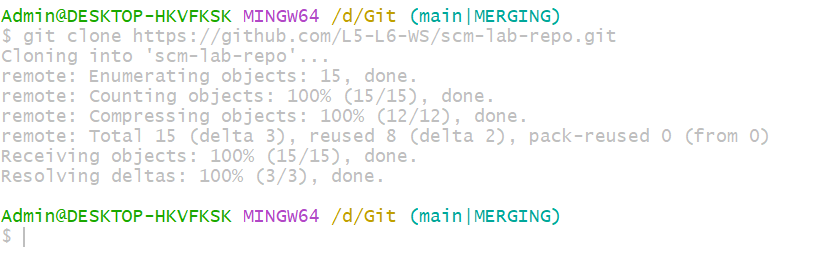


**Figure – 2**

**Step 2: Clone the Forked Repository Locally**

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

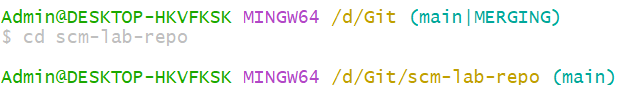
**git clone** [**https://github.com/your-username/Hello-World.git**](https://github.com/your-username/Hello-World.git)



**Figure – 3**

**Step 3: Change Directory to the Cloned Repo**

**cd scm-lab-repo**



**Figure – 4**

**Step 4: Add a New File or Modify Existing One**

**Vi hi.html**

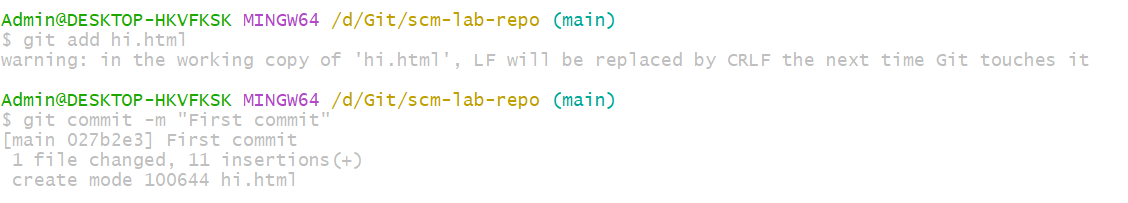


**Figure – 5**

**Step 5: Stage and Commit Your Changes**

**git add .**

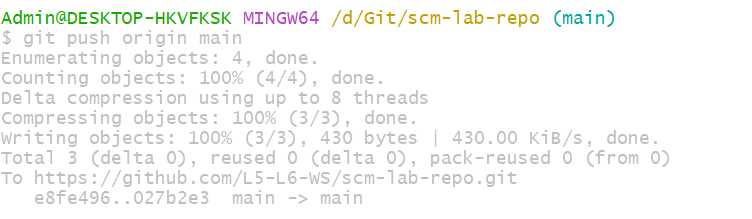
**git commit -m "First Commit"**

 **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**

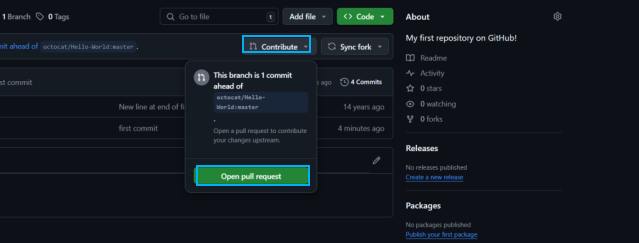


**Figure – 7**

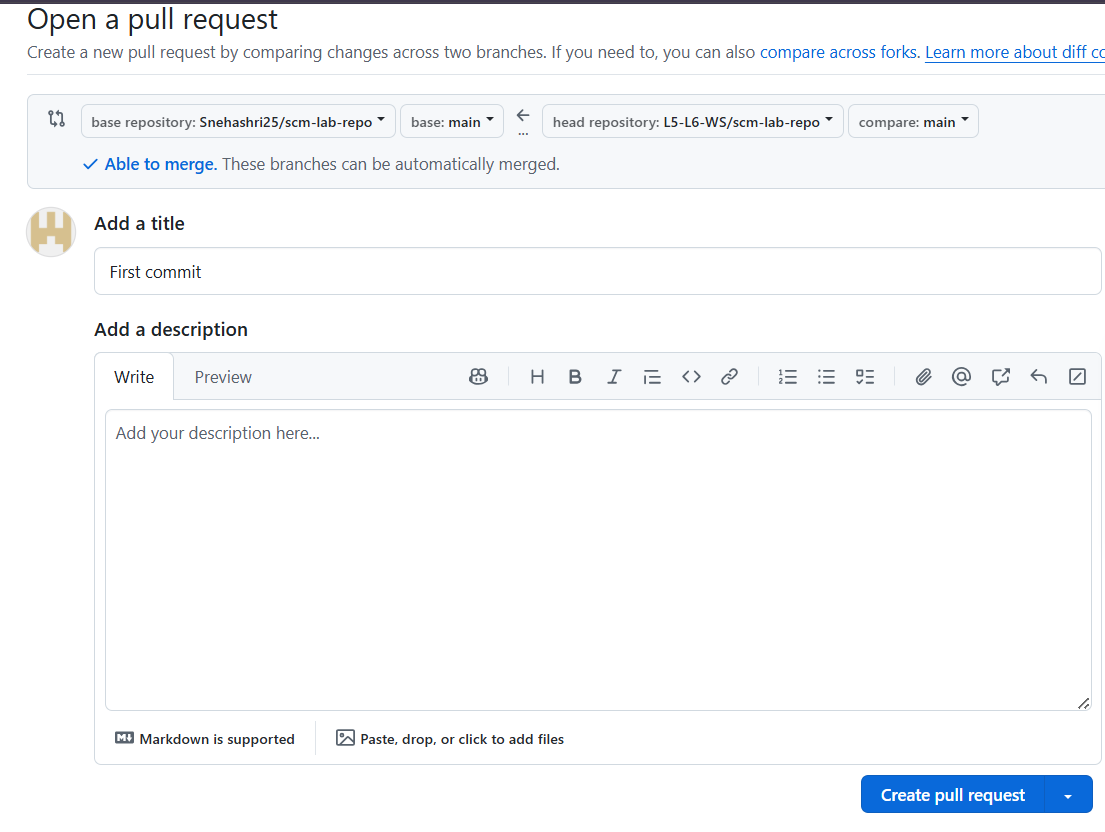
**Step 7: Create a Pull Request**

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.



**Figure - 8**

**Figure - 9**

**Source Code Management**

**LAB REPORT – 7**

**Step 1 : Checking all the commits in main branch and dev branch**

* Git log main --oneline

**Figure - 1**

* Git log dev --oneline

**Figure - 2**

* git log --oneline --all --graph --decorate

**Figure - 3**