

**Lab Session 1: Git Fundamentals**

**Computer**

A **computer** is any device capable of performing calculations, whether they are logical or mathematical.

**Program/Code**

A **program** (or **code**) is a set of instructions, often organized as an algorithm, that directs a computer to perform a specific task.

**Need for Managing Source Code**

Modern applications, such as Spotify, consist of multiple programs working together on both the frontend and backend to deliver smooth user experience. Regular updates are essential for:

* **Fixing Bugs:** Quickly resolving errors that may occur.
* **Improving UI/UX:** Enhancing the user interface and overall experience.
* **Optimizing Performance:** Addressing and refining issues for better performance.

For programmers, effective management of source code is crucial because:

* It ensures that all files remain in context throughout the lifecycle of the program.
* It facilitates collaboration, allowing multiple developers to work together on a shared codebase.

**Tools for Source Code Management**

**Git**

A version control system that runs locally on your computer. Git helps track changes and manage versions of your project.

**GitHub**

A global, cloud-based platform that hosts Git repositories, enabling developers to share, collaborate, and contribute to projects from anywhere in the world.

**Version**

A **version** in version control represents a snapshot of your project at a specific moment in time. This snapshot allows you to review, revert, or compare changes made throughout the development process.

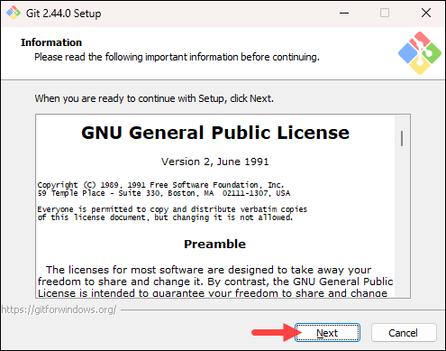
**GIT INSTALLATION**

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

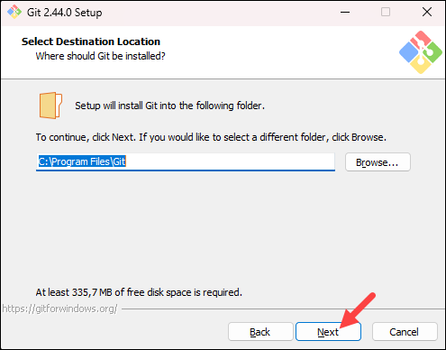


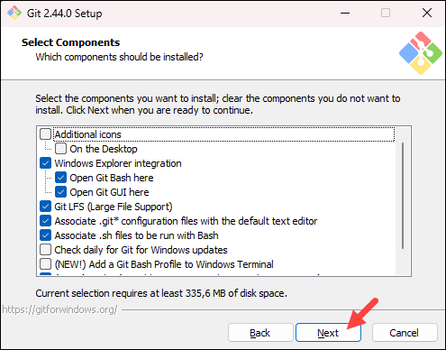
2. Double-click the downloaded [**file**](https://phoenixnap.com/glossary/what-is-a-file) to extract and launch the installer

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

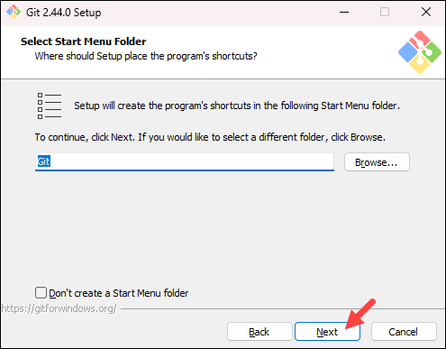


4.The installer prompts you for an installation location. Leave the default one unless you want to change it, and click **Next**.



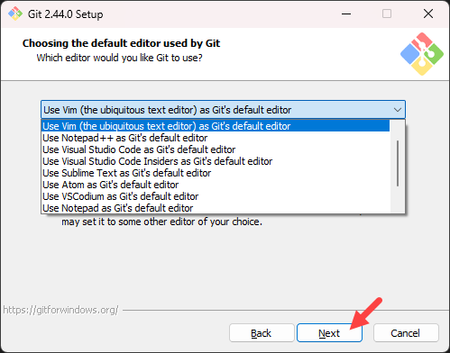
5.In the component selection screen, leave the defaults unless you need to change them and click **Next**.

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

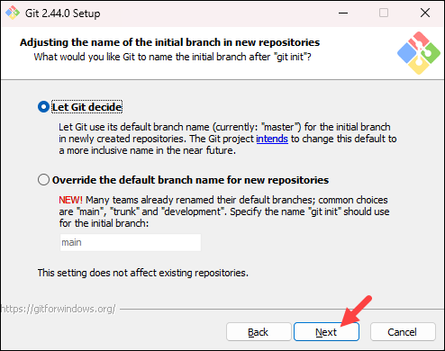


7. Select a text editor you want to use with Git. Use the drop-down menu to select Notepad++ (or whichever text editor you prefer) and click **Next**.

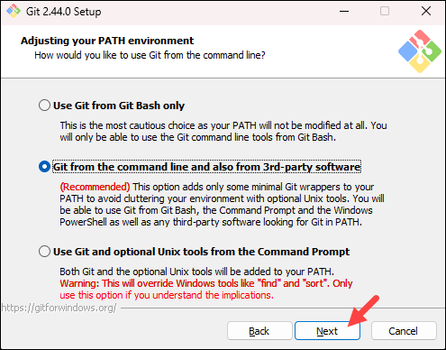
If you prefer to use a CLI text editor in [**Git Bash**](https://phoenixnap.com/kb/what-is-git-bash), select [**nano**](https://phoenixnap.com/kb/use-nano-text-editor-commands-linux) or [**Vim**](https://phoenixnap.com/kb/vim-commands-cheat-sheet) from the list.



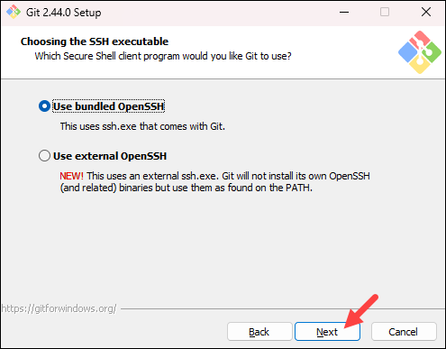
8. The next step allows you to choose a different name for your initial branch. The default is **master**. Unless you are working in a team that requires a different name, leave the default option and click **Next.**



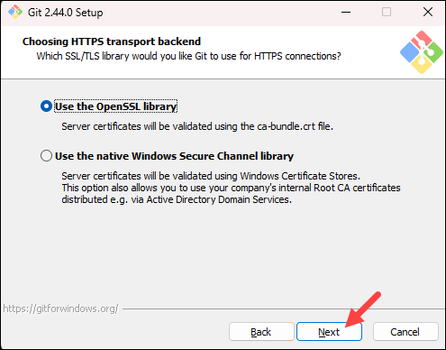
9. The next step allows you to change the **PATH environment**. The **PATH**is the default set of [**directories**](https://phoenixnap.com/glossary/what-is-a-directory) included when you run a command from the command line. Keep the middle (recommended) selection and click **Next**.



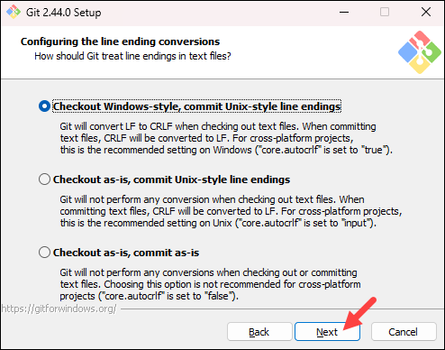
10. The installer prompts you to select the SSH client for Git to use. Git already comes with its own SSH client, so if you don't need a specific one, leave the default option and click **Next.**



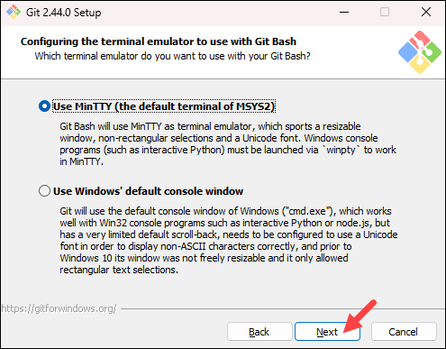
11. The next option relates to server certificates. The default option is recommended for most users. If you work in an Active Directory environment, you may need to switch to Windows Store certificates. Select your preferred option and click **Next**.



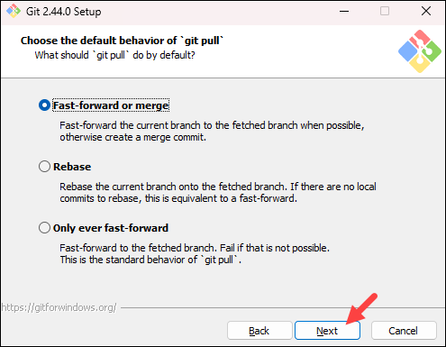
12. The following selection configures line-ending conversion, which relates to the way data is formatted. The default selection is recommended for Windows. Click **Next** to proceed.



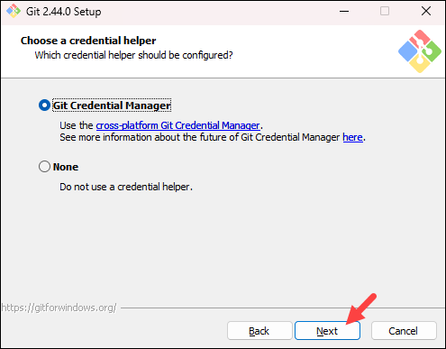
13. Choose the [**terminal emulator**](https://phoenixnap.com/glossary/terminal-emulation) you want to use. The default MinTTY is recommended for its features. Click **Next** to continue.



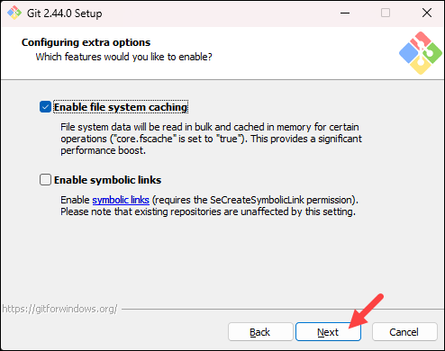
14. The next step allows you to choose what the **git pull** command will do. The default option is recommended unless you specifically need to change its behavior. Click **Next**to continue with the installation.



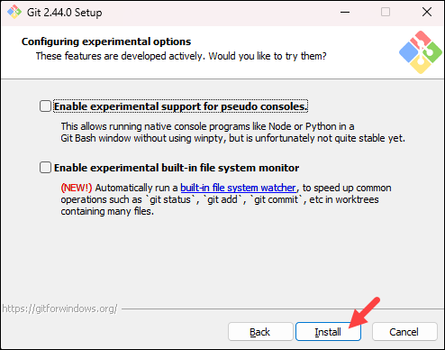
15. The next step is to choose which credential helper to use. Git uses credential helpers to fetch or save credentials. The default option is the most stable one. Select your preferred credential manager and click **Next**.



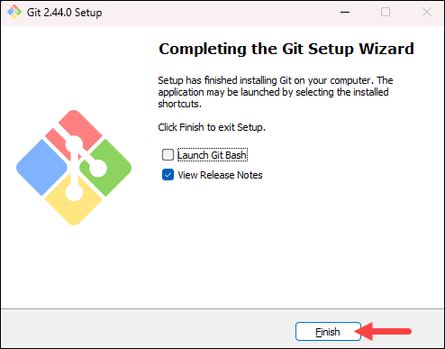
16. The next step lets you decide which extra options to enable. If you use [**symbolic links**](https://phoenixnap.com/kb/symbolic-link-linux), which represent shortcuts for the command line, tick the box. Keep [**file system**](https://phoenixnap.com/glossary/filesystem) caching checked and click **Next**.



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



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



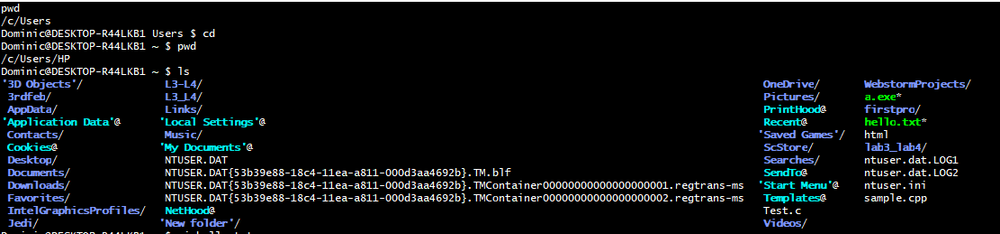
**COMMNADS**

**1.pwd**

* Presenting work directory
* used to view the current directory working on

**2.cd**

* change directory
* used to change the directory

**3.ls**

* used to list all the folders

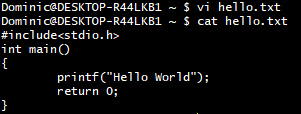
**4.vi**

* vim mode/vi editor
* used to create a file, edit it
* Syntax: vi nameoffile (Ex: vi hello.txt)

click the key i to insert or edit your file. then esc to return from insert mode :wq(then enter) to return

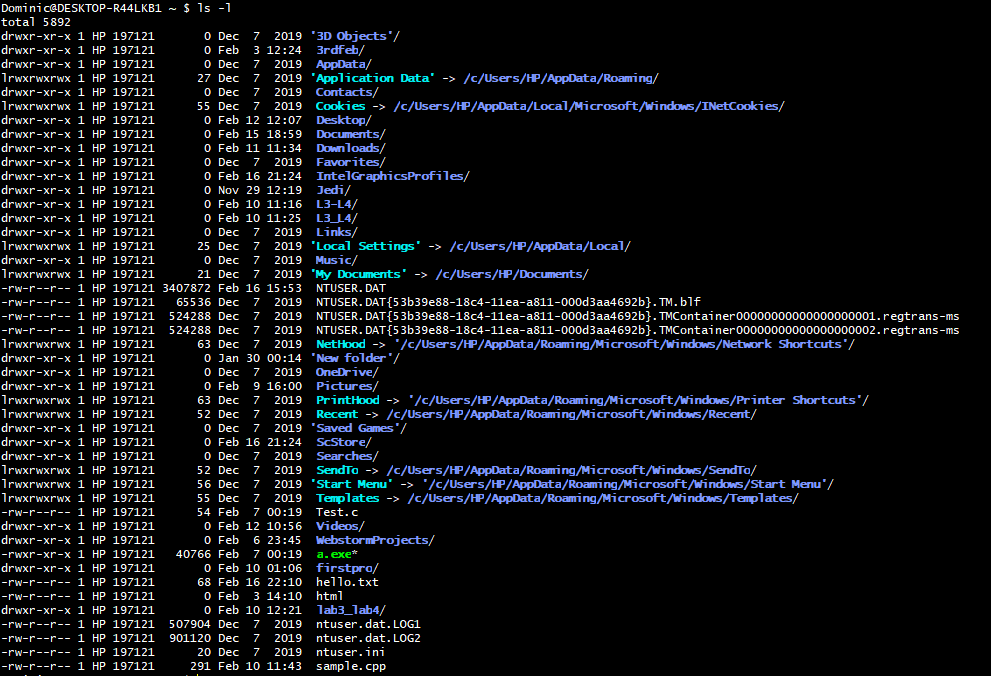
**5. cat**

* syntax: cat filename (cat hello.txt)
* to view the file contents.

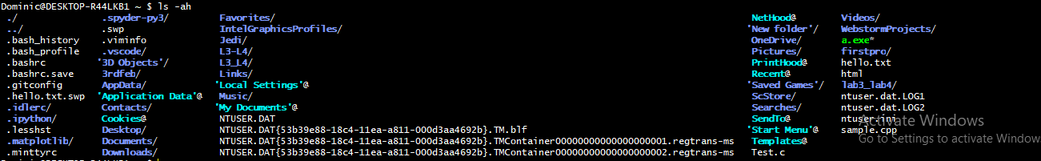


**6.ls -l**

* used to list all files

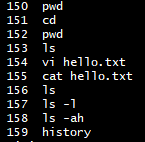


**7.ls -ah**



**8.history**

* used to view history of commands



**9. mkdir**

* used to create folder
* Syntax: mkdir nameoffolder

:ls nameoffolder

* confirmation to check if the folder is created

:cd folder/

* to change the present directory as the folder

7ea2f5493866af804c5f544cd865de46.png

* also after this create one new file by using vi (then insert i)

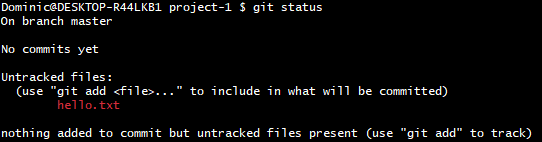
**10.git init**

* initialize the repository
* working directory to empty repository

c091cd6d7ffe7a4bc143c3b20c6030e1.png

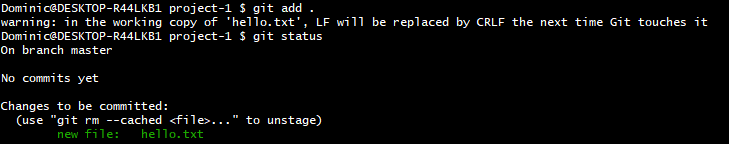
**11.git status**

* display the current status of the working directory.



**12.git add file**

* to add the file
* Syntax: git add filename
* After this edit the file (using vi, insert i)



**13. git commit**

* Used to commit a change
* Syntax: git commit -m “Message”

f4e9c7f0f1dae2e11a6979933ede1e33.png

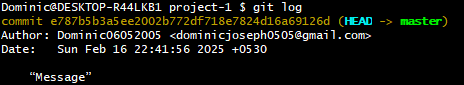
**14. git config**

* Used to configure username and email
* syntax:

0b920618f6ed052af5d179c2265107e0.png

**15. git log**

* used to view status/entries of commits



**16. git diff**

* shows the changes made in the working directory since the last commit

**17.git branch<branch name>**

* creates a new branch

5a2f0db2c5e6812b941d8bd85f962be8.png

**18. git branch**

* lists all the branches in the repository

6b3e0990f5138ea3ca6be9db6a3f65e2.png

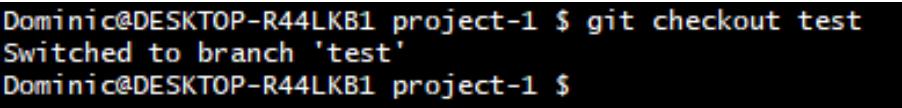
**19.git log -onelinegit**

* view a concise commit history

06de73da4459d7ba0506ac92260dc414.png

**20. git checkout<branch name>**

* switch to the master branch



**SCM Project**

The project was to make a repository in GitHub, make 3 branches and merge it with the main branch and access all 4 team-mate’s repositories, fork it, clone it, make some changes and merge them.

First, make your own repositories and make 3 branches and add files and merge with the main branch.

1)Go to the directory on your computer



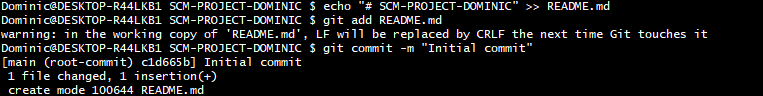
2)Clone the GitHub repositories



3)Change to the folder



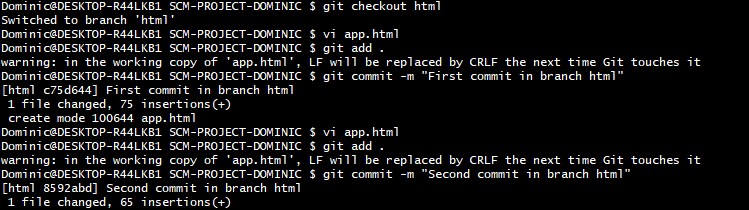
4)Make an initial commit as a README.md and commit it



5)Make four branches A black background with white text

AI-generated content may be incorrect.

6)Checkout to the first branch, add the files and make two commits



7)Checkout to the second branch, add the files and make two commits A screen shot of a computer

AI-generated content may be incorrect.

8)Checkout to the third branch, add the files and make two commits A screen shot of a computer program

AI-generated content may be incorrect.

9)Checkout to the main branch



10)Merge all the branches to the main branch

A screenshot of a computer program

AI-generated content may be incorrect.

11) Push all the commits and files in the Github repository

A black screen with white text

AI-generated content may be incorrect.

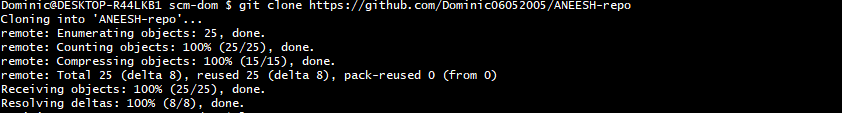
*\*(Now we will clone the repository of the rest of my Team Members, make some changes in their repositories, and send pull requests.)*

* Aneesh

1)Go back to the folder where you want to save the folder



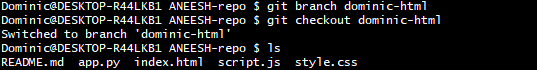
2)Clone the forked repository



3)Go to the cloned repository



4)Create a new branch and checkout to it



5)Make some changes in the repository.



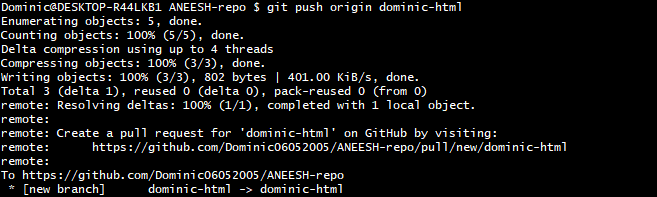
6)Add the edited files.



7)Commit the files.



8)push it to the Forked Repository for pull request.

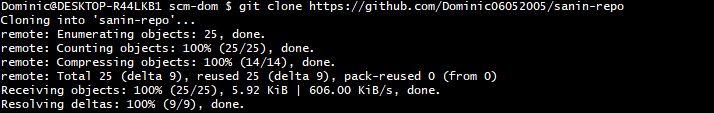


* Sanin

1)Go back to the folder where you want to save the folder



2)Clone the Forked repository



3)Go to the cloned repository.



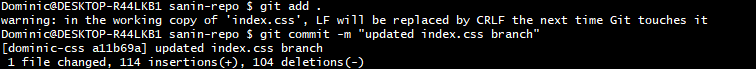
4)Create a new branch and checkout to it.



5)Make changes in the file



6)Add the files and commit the files.



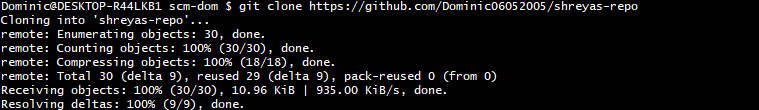
7)Push the branch to the forked repository for pull request.



* Shreyas

1)Go back to the folder where you want to save the folder 

2)Clone the Forked Repository.



3)Go to the cloned repository.



4)Create a branch and checkout to it.



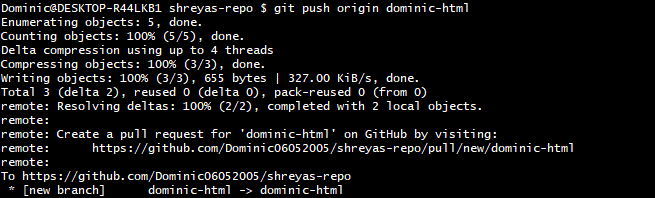
5) Make changes to the files.



6)Add the files and then commit it.



7)Push the branch to the forked repository for pull request.



*\*(Finally, after all the changes made and pull requests we use “git pull” to sync all the changes in our local folder)*