Subject Name: **Source Code Management**

Subject Code: **CS181**

Cluster: **Beta**

Department: **CSE**

Submitted By:
Dilpreet Singh
2110990452
G-8

Submitted To:
Mr. Monit Kapoor

CHITKARA UNIVERSITY

Source Code Management (Task 1.1)

CHITKARA UNIVERSITY

Setting Up Git Client

Git is a widely used open-source software tracking application used to track projects across different teams and revision levels.

This guide will show us how to install Git on Windows.

Prerequisites :

- Administrator privileges
- Access to a command-line
- Your favourite coding text editor
- Username and password for the GitHub website (optional)

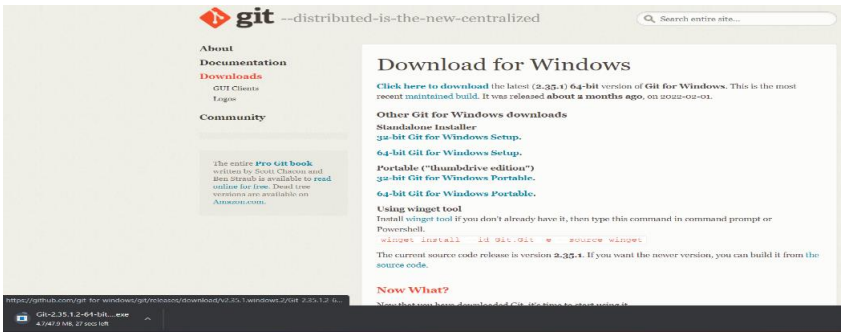
Steps For Installing Git for Windows

Installing Git prompts you to select a text editor. If you don't have one, we strongly advise you to install prior to installing Git. Our roundup of the best text editors for coding may help you decide.

Download Git for Windows

1 . Browse to the official Git website: https://git-scm.com/downloads

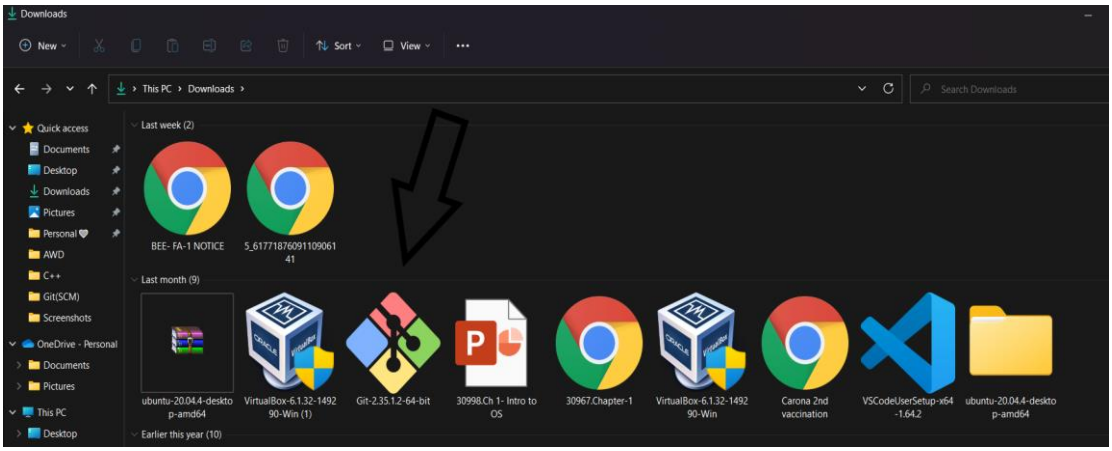
2. Click the download link for Windows and allow the download to complete



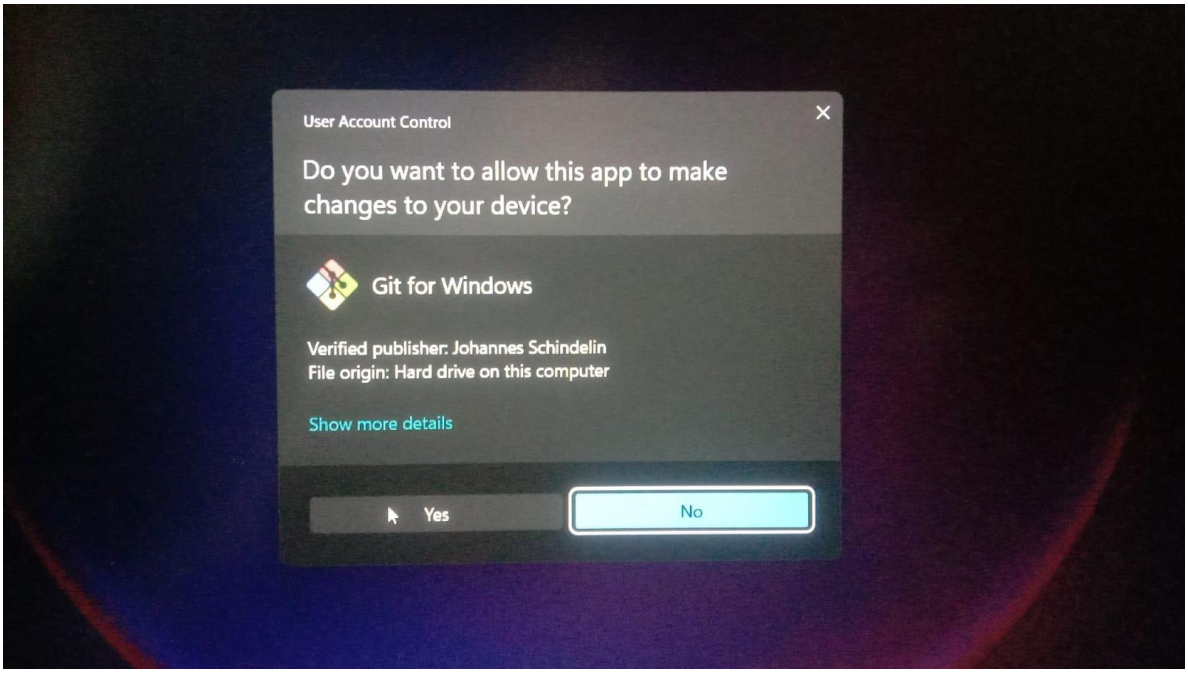
CHITKARA UNIVERSITY

Extract and Launch Git Installer

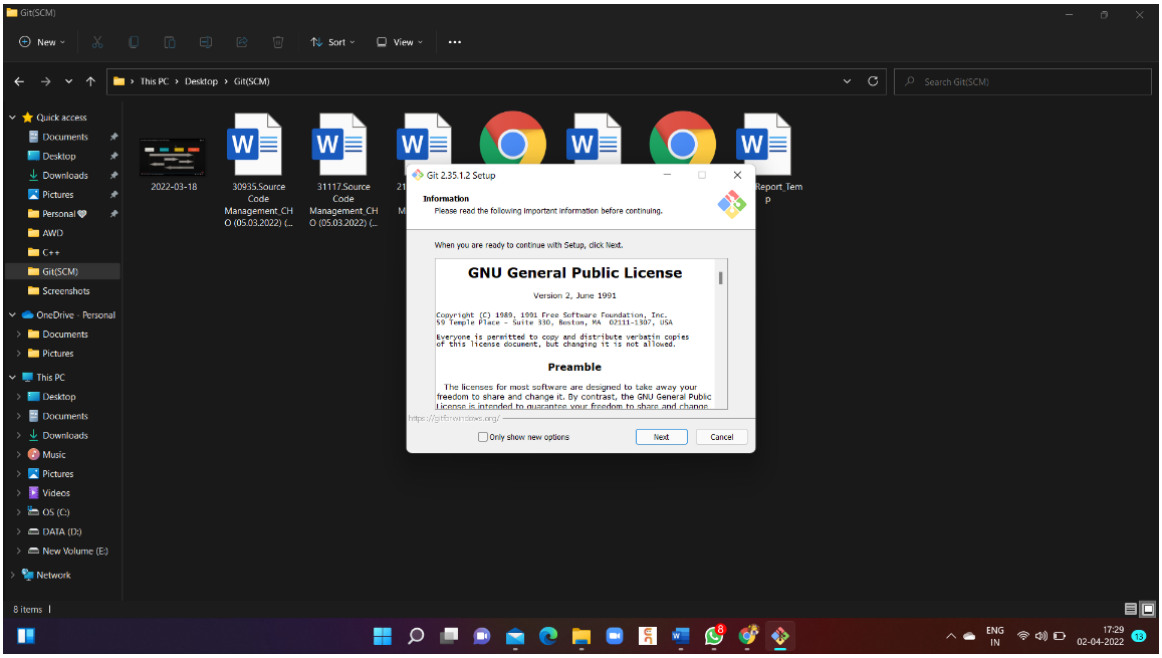
3. Browse to the download location (or use the download shortcut in your browser). Double click thfile to extract and launch the installer.



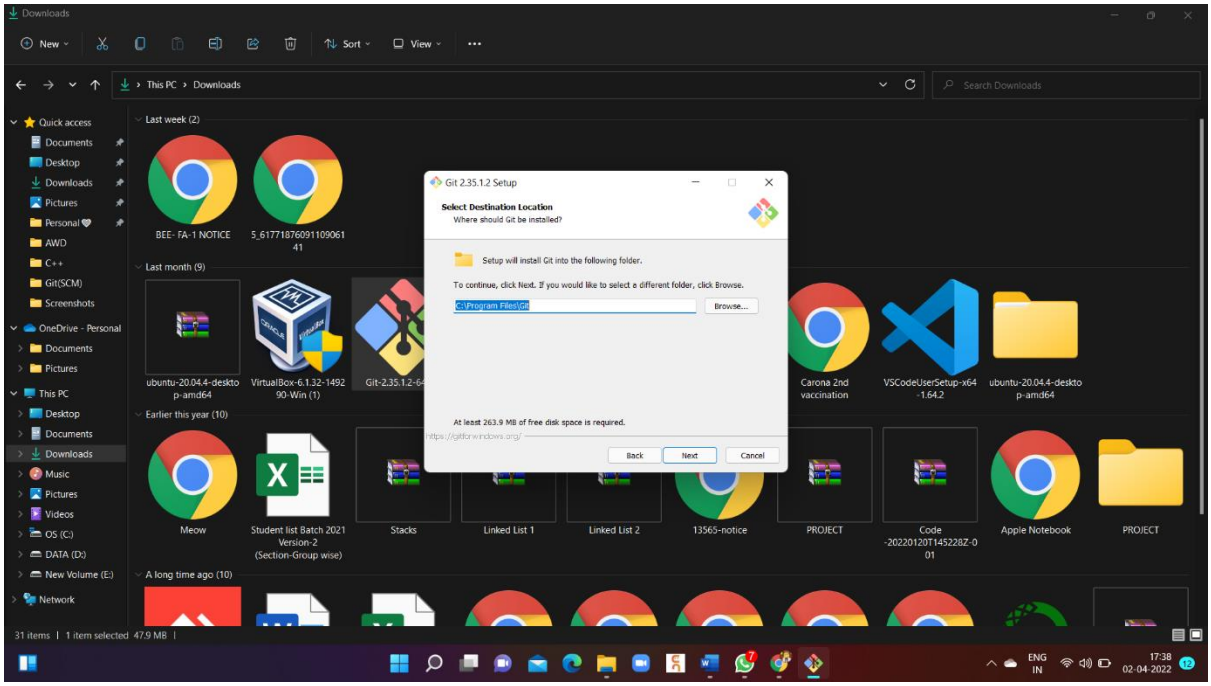
4.Allow the app to make changes to your device by clicking Yes on the User Account Control dialog that opens.



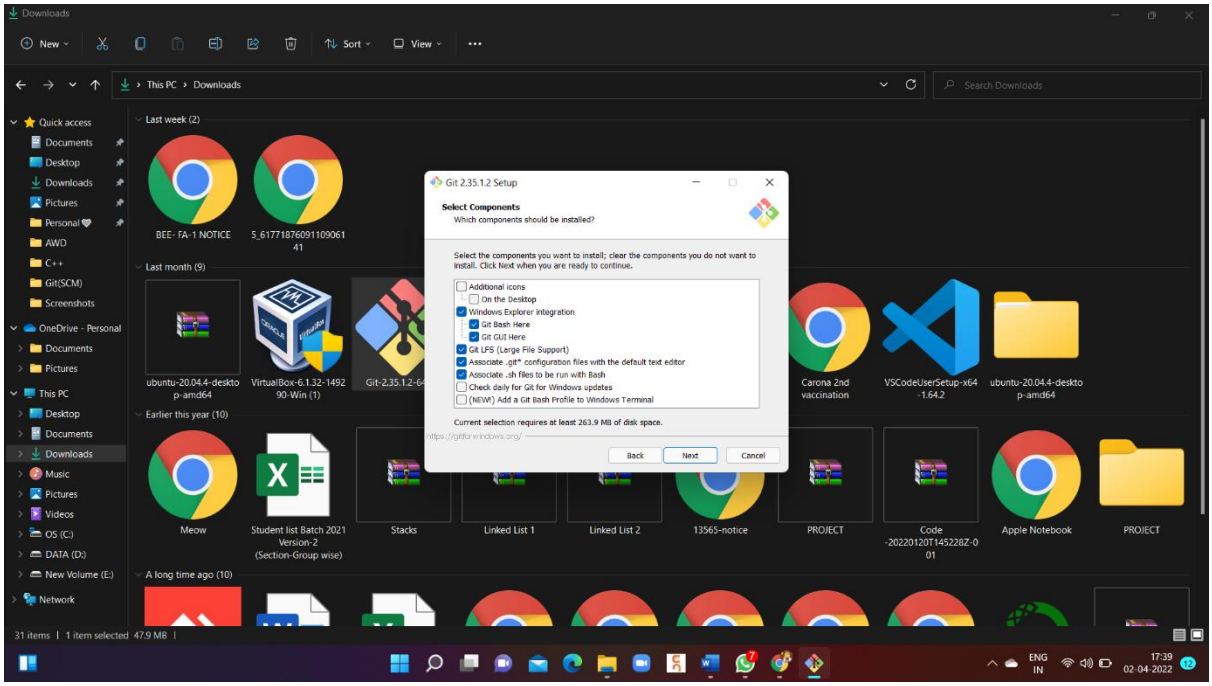
5.Review the GNU General Public License, and when you're ready to install, click Next.



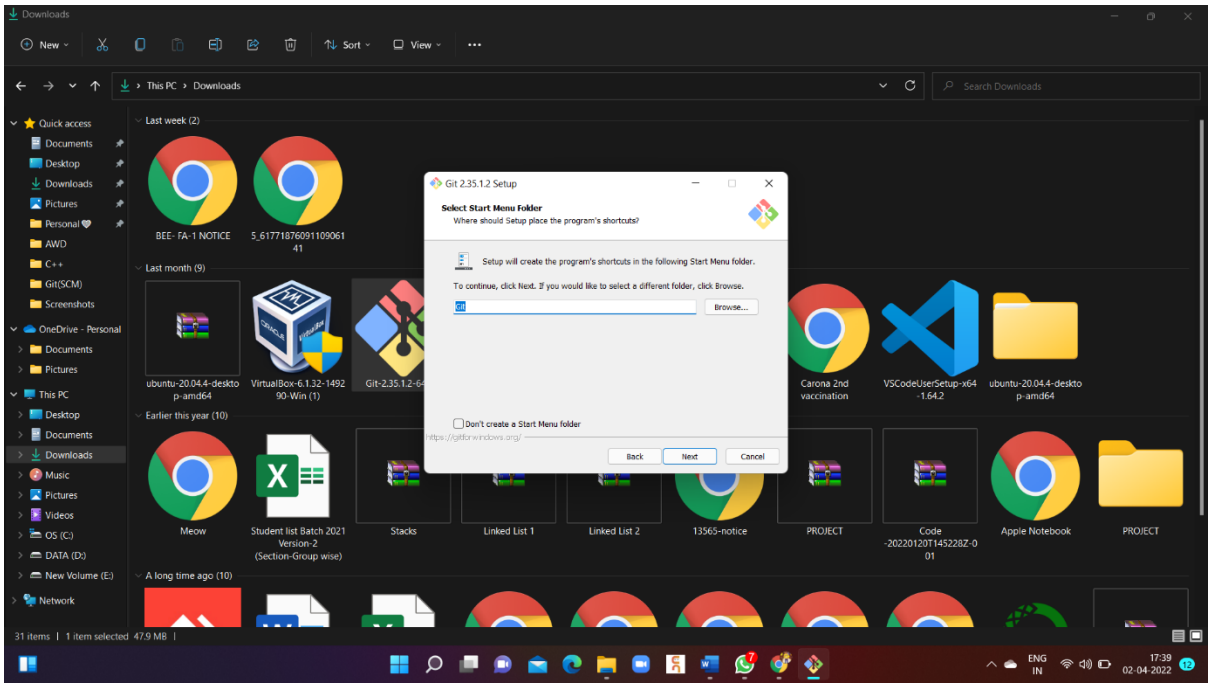
6.The installer will ask you for an installation location. Leave the default, unless you have reason to change it, and click **Next**.



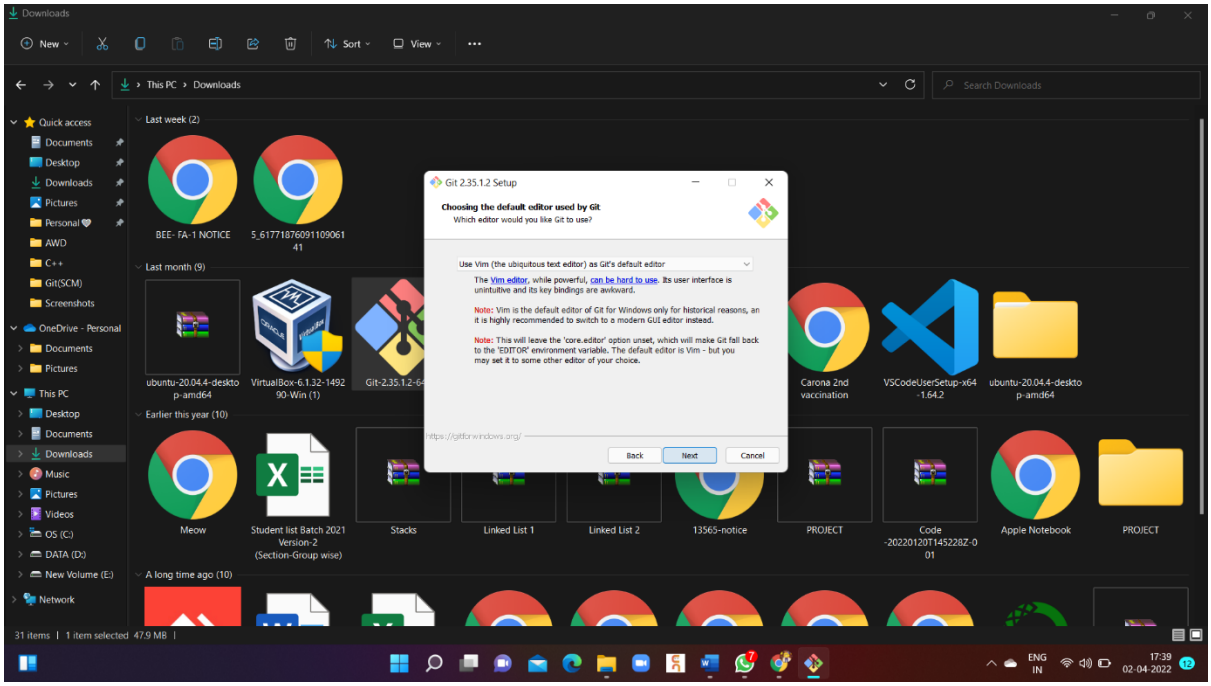
7. A component selection screen will appear. Leave the defaults unless you have a specific need to change them and click **Next**.



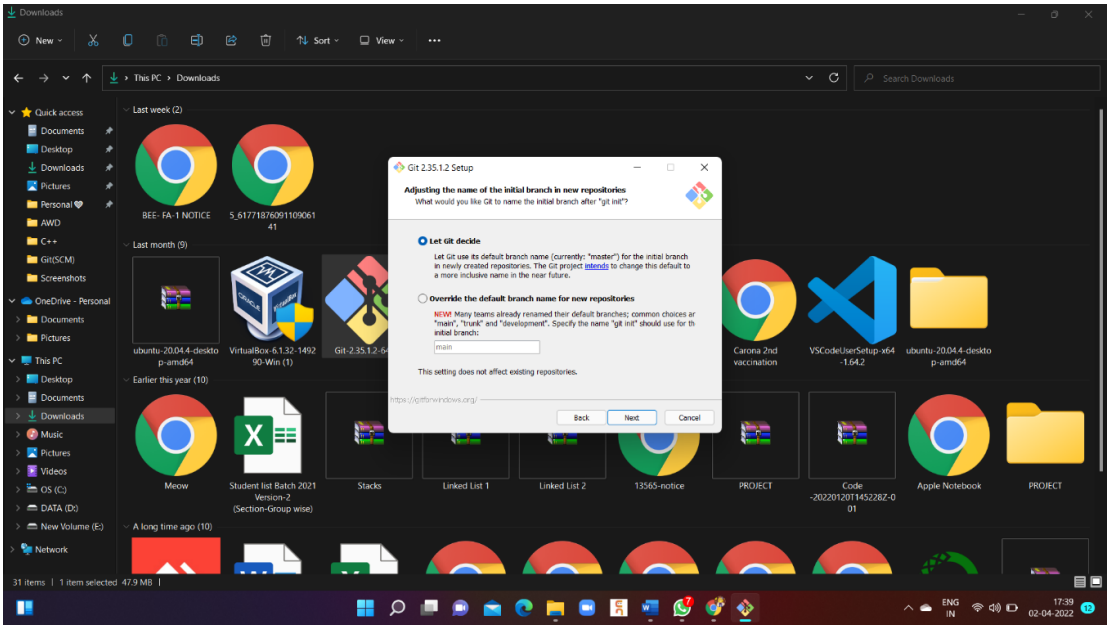
8. The installer will offer to create a start menu folder. Simply click Next.



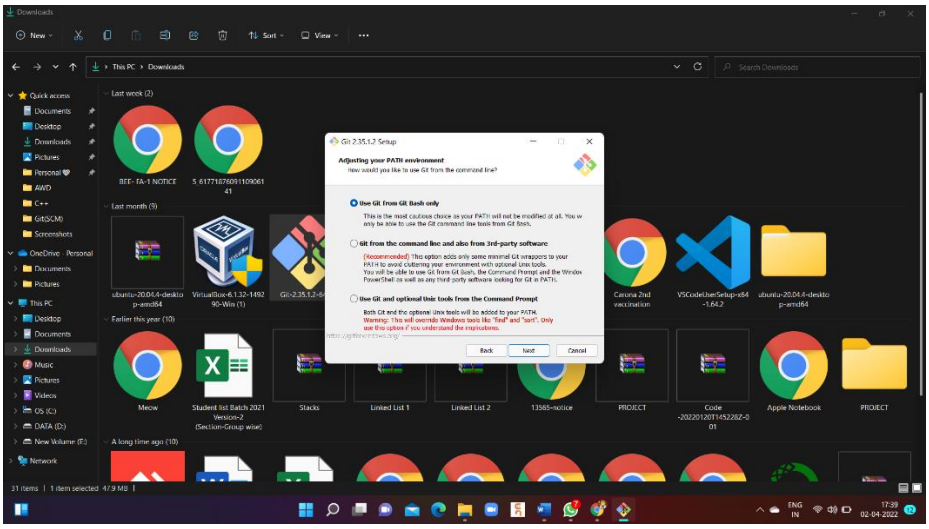
9.Select a text editor you'd like to use with Git. Use Vim (or whichever text editor you prefer) and click Next.



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

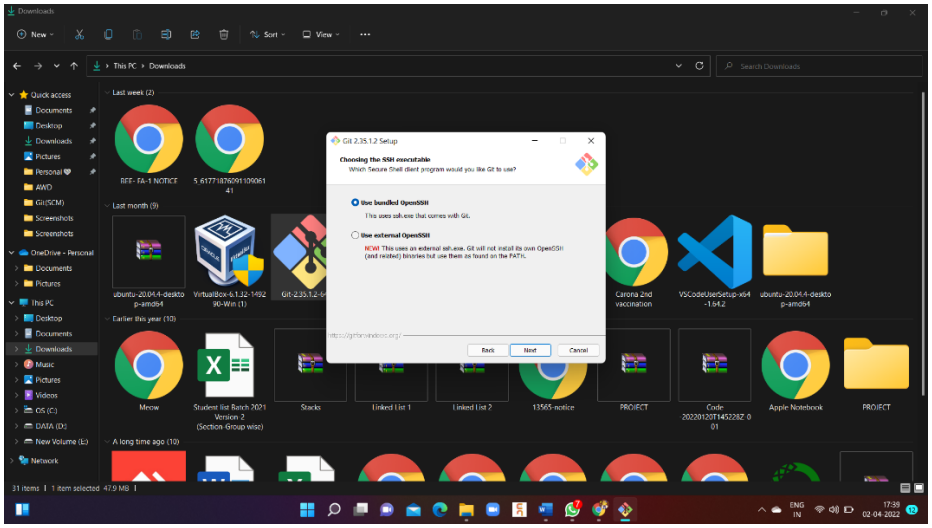


11. This installation step allows you to change the PATH environment. The PATH is the default set of directories included when you run a command from the command line. Change it to Use Git from Git Bash. Click Next

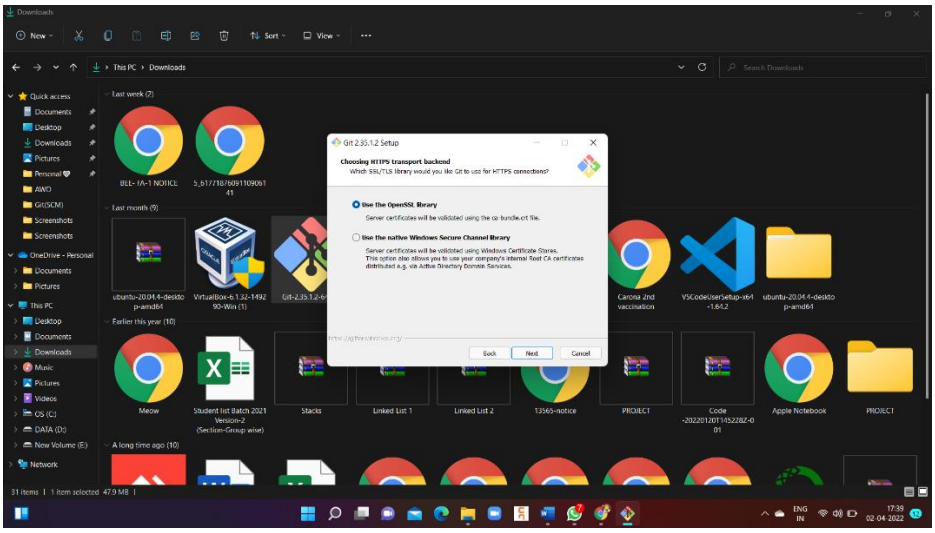


Server Certificates, Line Endings and Terminal Emulators

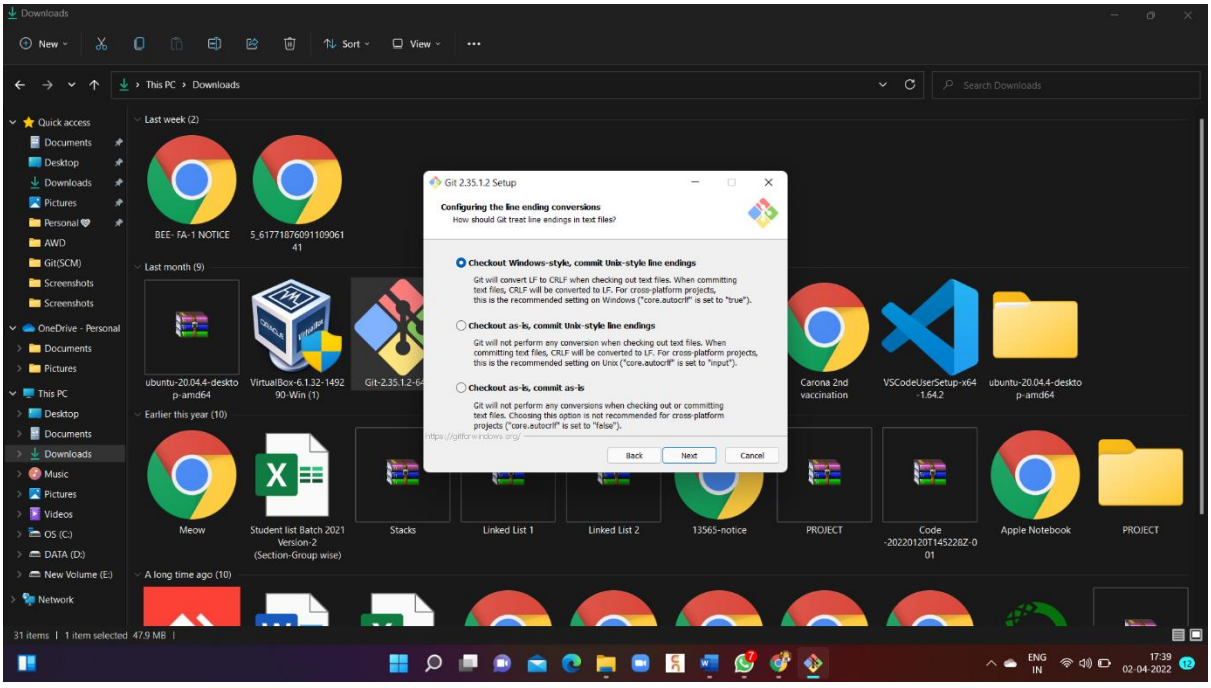
12. The installer now asks which SSH client you want Git to use. Git already comes with its own SSH client, so if you don't need a specific one, leave the default option and click Next.



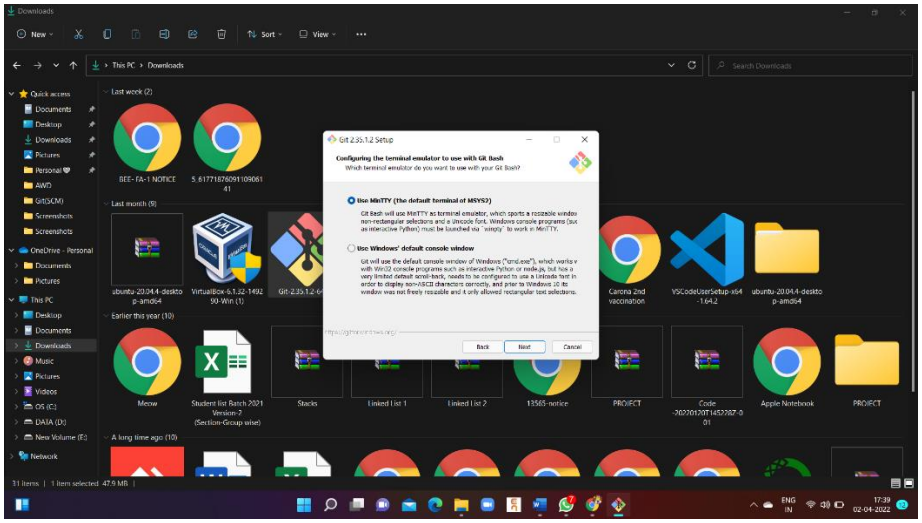
13. The next option relates to server certificates. Most users should use the default. If you're working in an Active Directory environment, you may need to switch to Windows Store certificates. Click Next.



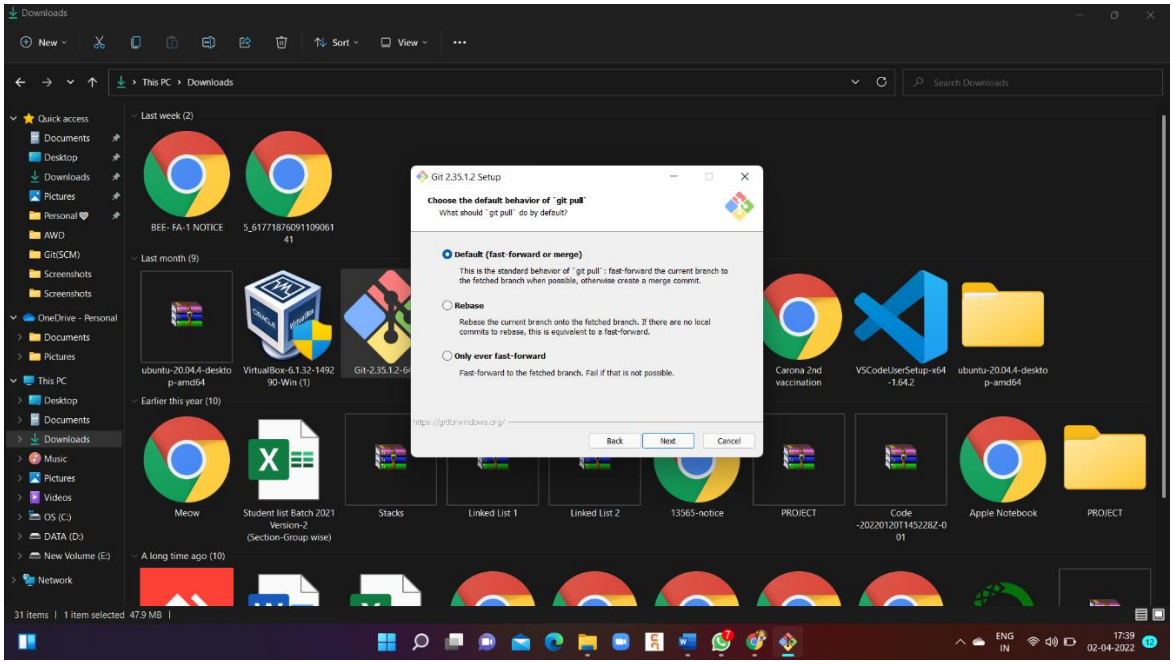
14. The next selection converts line endings. It is recommended that you leave the default selection. This relates to the way data is formatted and changing this option may cause problems.



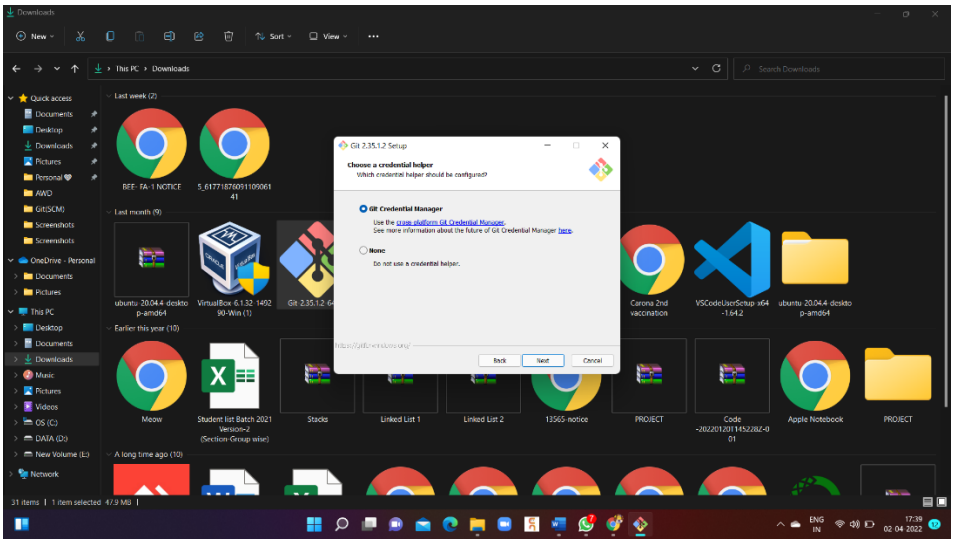
15. Choose the terminal emulator you want to use. The default MinTTY is recommended, for its features. Click **Next.**



16. The installer now asks what the git pull command should do. The default option is recommended unless you specifically need to change its behaviour. Click Next to continue with the installation.

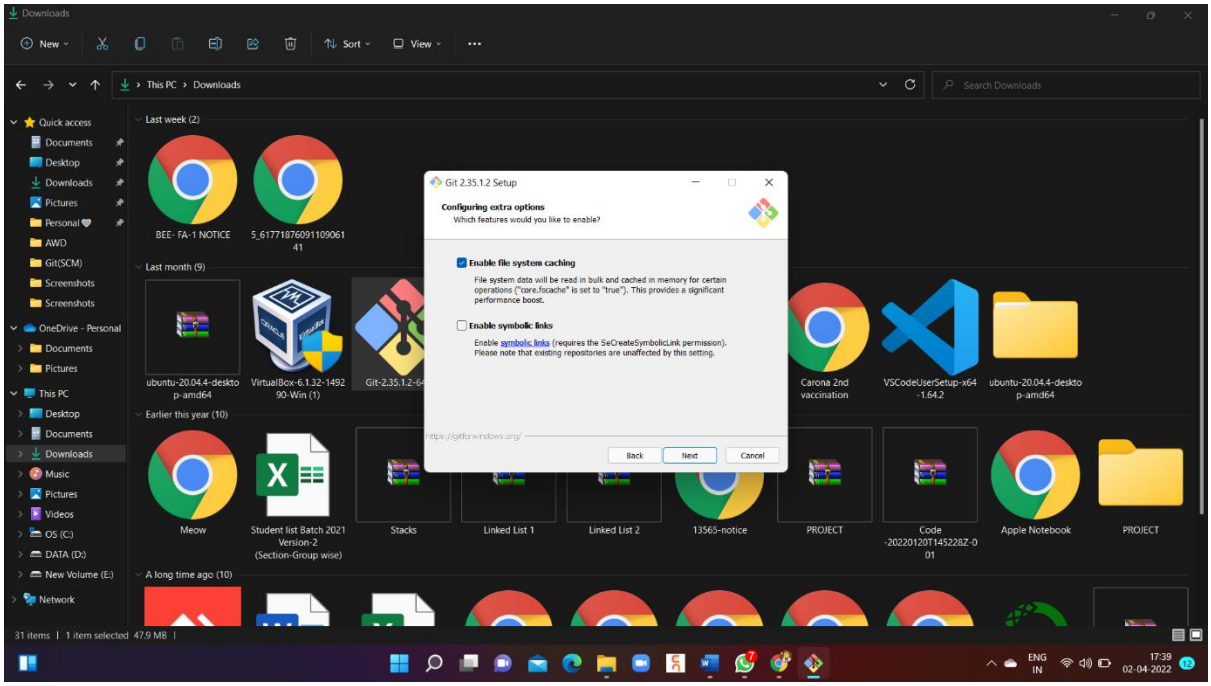


17. Next you should choose which credential helper to use. Git uses credential helpers to fetch or save credentials. Leave the default option as it is the most stable one, and click **Next.**

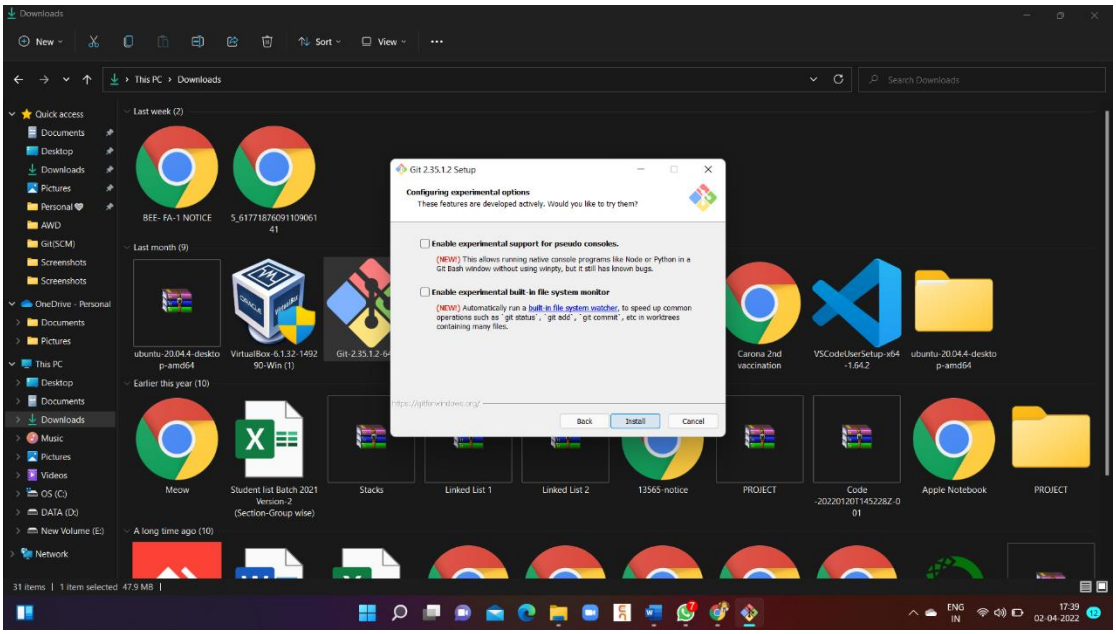


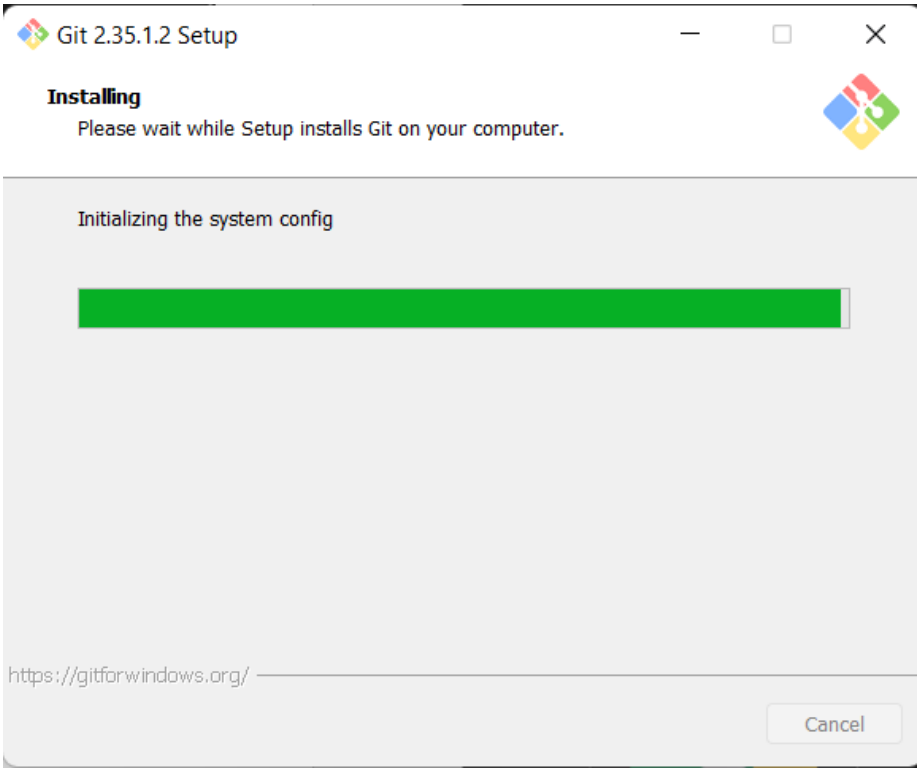
Additional Customization Options

18.The default options are recommended; however this step allows you to decide which extra option you would like to enable. Choose Enable File System Caching. Click **Next**.

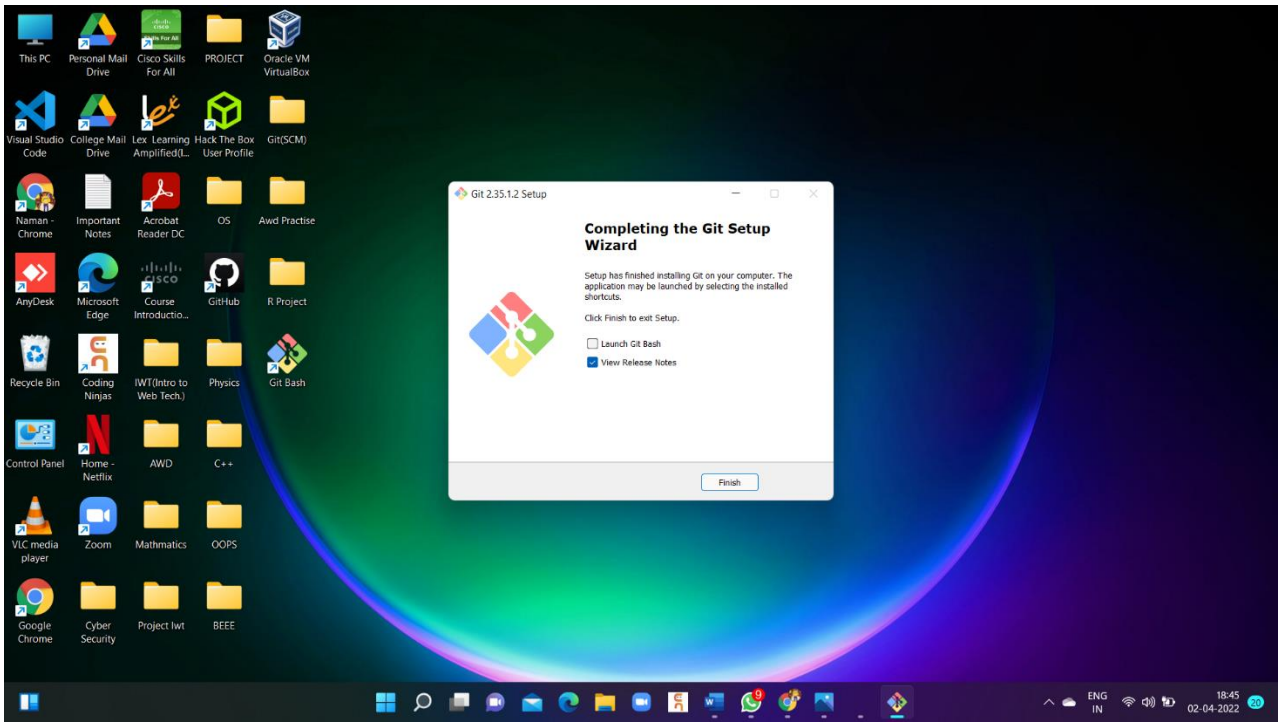


19. Depending on the version of Git you're installing, it may offer to install experimental features. At the time this article was written, the options to include support for pseudo controls and a built-in file system monitor were offered. Unless you are feeling adventurous, leave them unchecked and click **Install.**





20. Once the installation is complete, tick the boxes to view the Release Notes or Launch Git Bash, then click Finish



CHITKARA UNIVERSITY

Setting Up GitHub Account

Part 1: Configuring your GitHub account

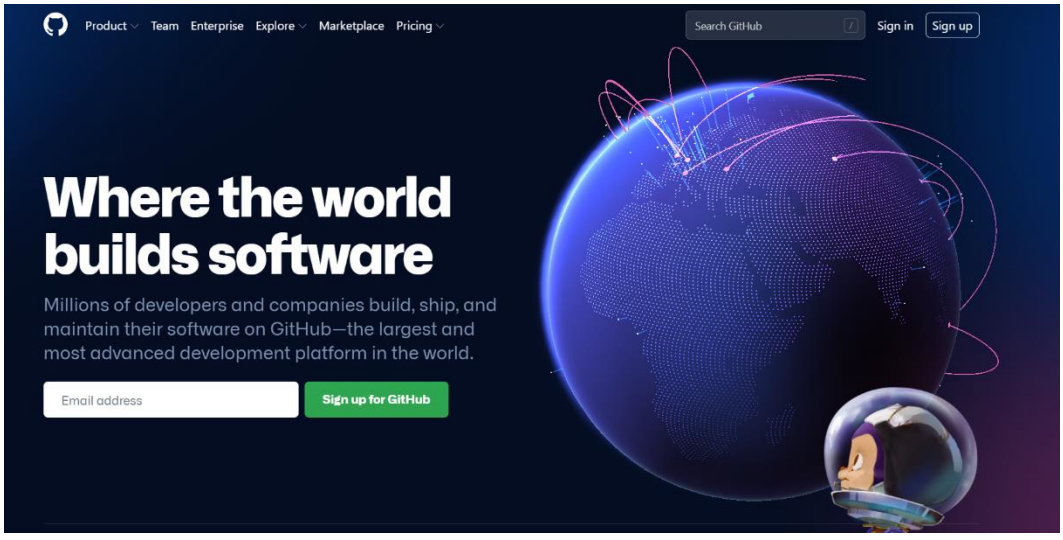
The first steps in starting with GitHub are to create an account, choose a product that fits your needs best, verify your email, set up two-factor authentication, and view your profile.

There are several types of accounts on GitHub. Every person who uses GitHub has their own user account, which can be part of multiple organizations and teams. Your user account is your identity on GitHub.com and represents you as an individual.

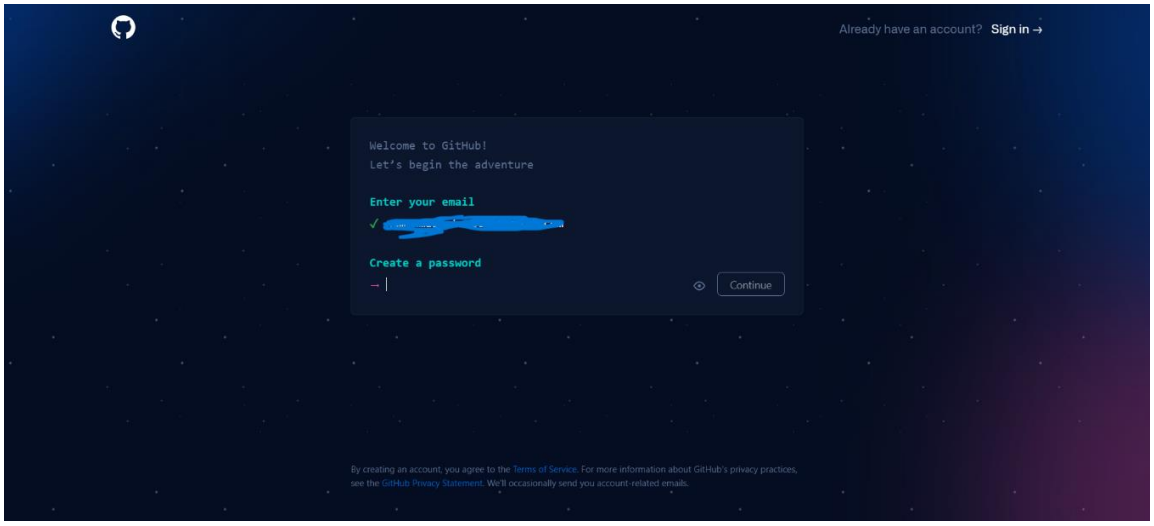
1. Creating an account

To sign up for an account on GitHub.com, navigate to https://github.com/ and follow the prompts.

Enter Your Email to Sign up for Git Hub.



To keep your GitHub account secure you should use a strong and unique password. For more information, see "Creating a strong password."



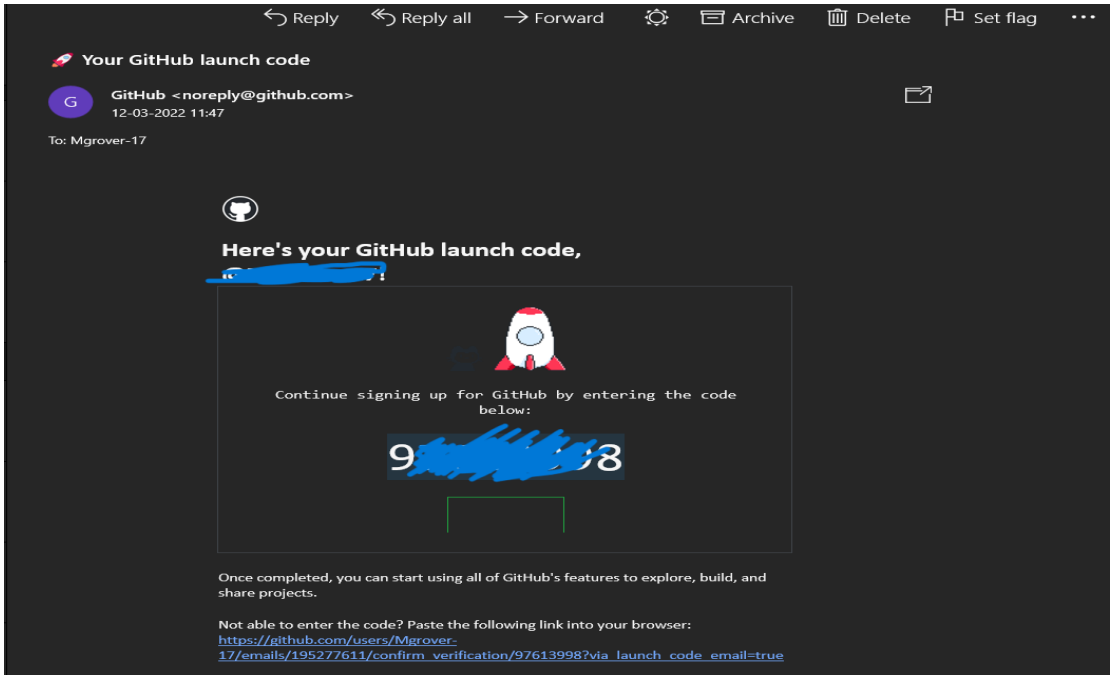
2. Choosing your GitHub product

You can choose GitHub Free or GitHub Pro to get access to different features for your personal account. You can upgrade at any time if you are unsure at first which product you want.

For more information on all of GitHub's plans, see "GitHub's products."

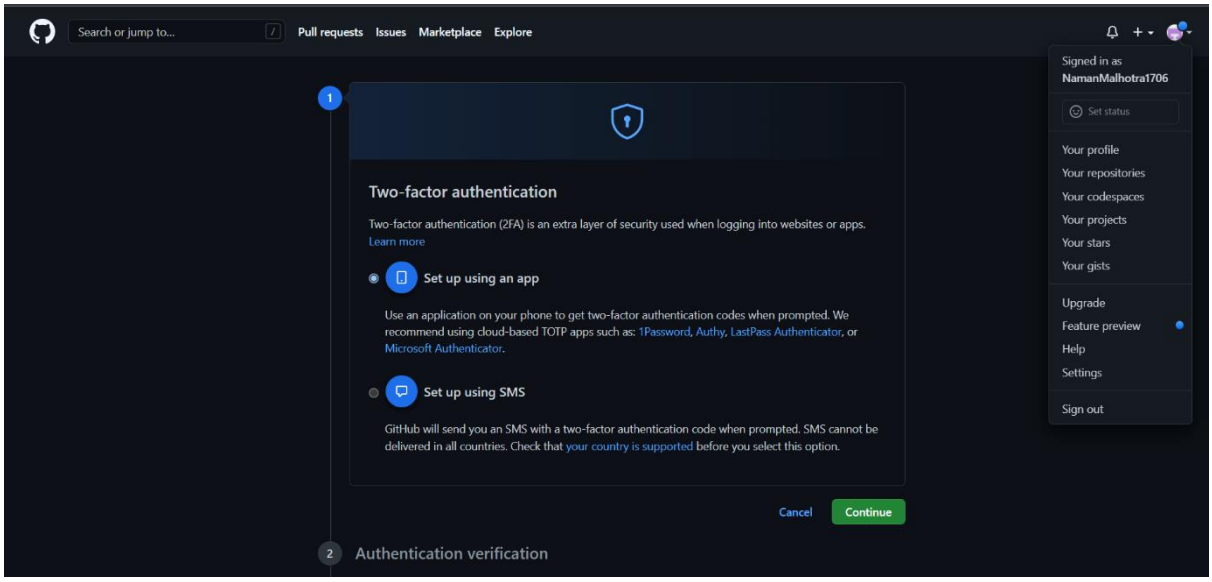
3. Verifying your email address

To ensure you can use all the features in your GitHub plan, verify your email address after signing up for a new account. For more information, see "Verifying your email address."



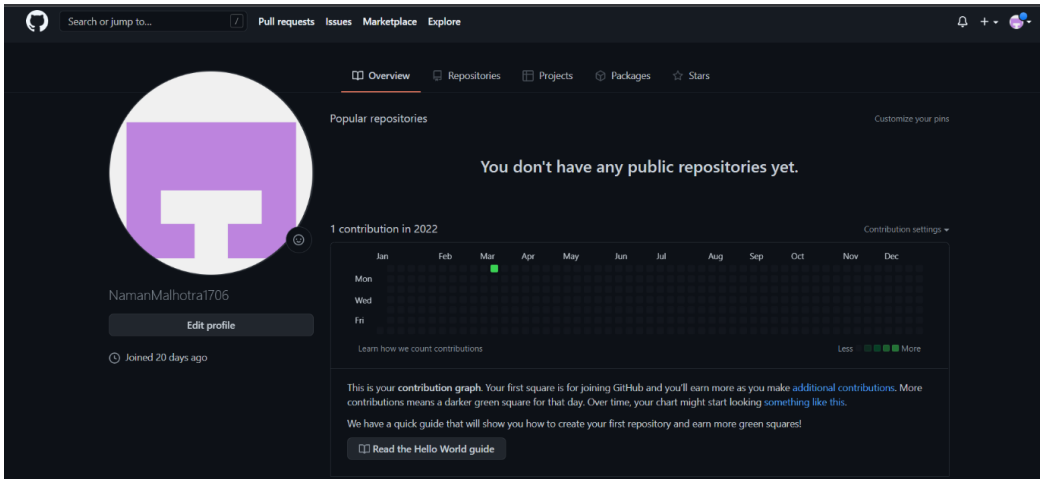
4. Configuring two-factor authentication

Two-factor authentication, or 2FA, is an extra layer of security used when logging into websites or apps. We strongly urge you to configure 2FA for the safety of your account. For more information, see "About two-factor authentication."



5. Viewing your GitHub profile and contribution graph

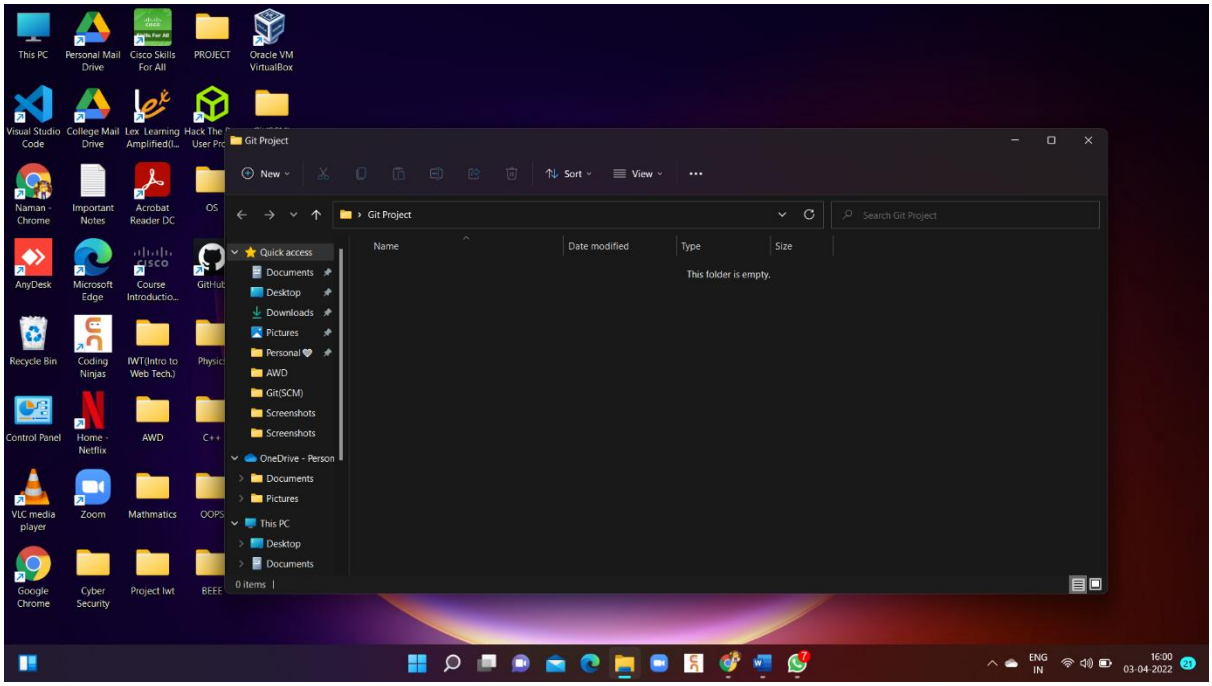
Your GitHub profile tells people the story of your work through the repositories and gists you've pinned, the organization memberships you've chosen to publicize, the contributions you've made, and the projects you've created. For more information, see "About your profile" and "Viewing contributions on your profile."



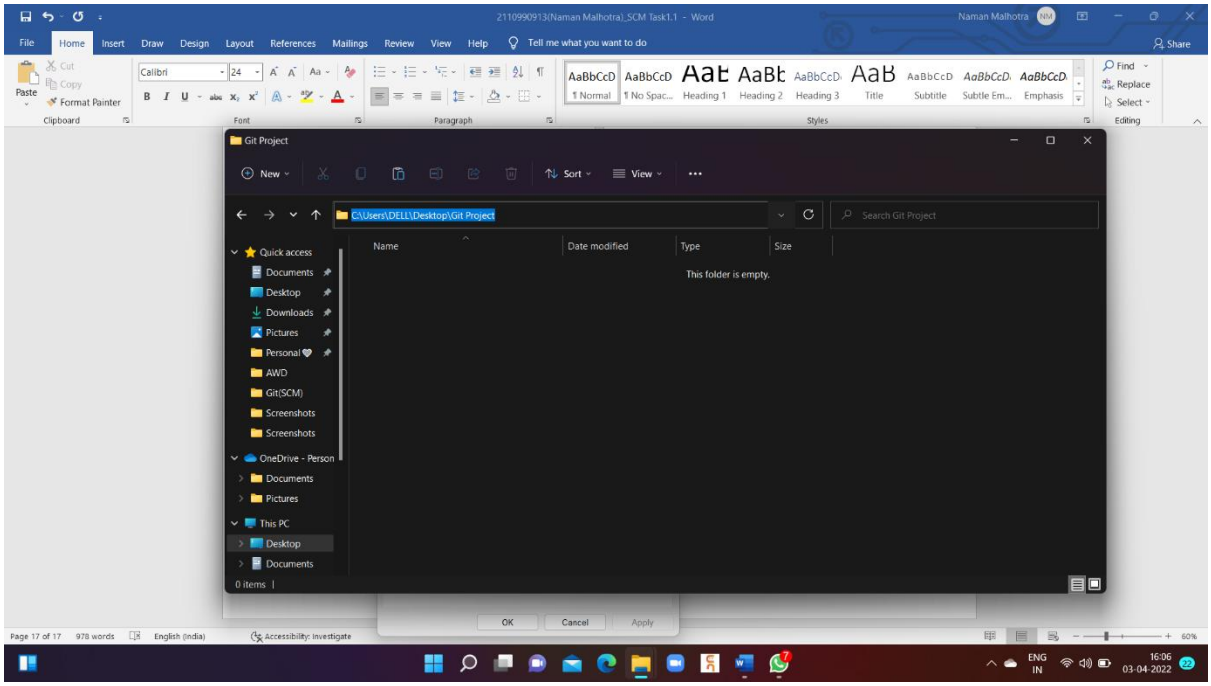
CHITKARA UNIVERSITY

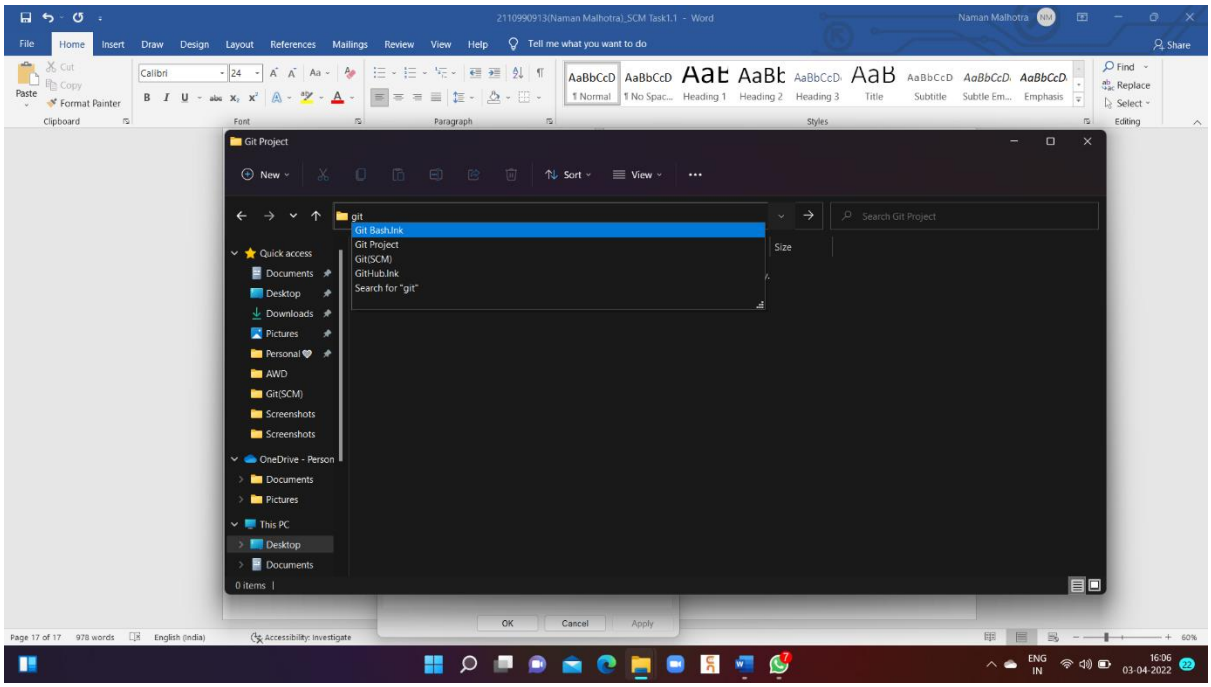
Generates Log

1. Create a New Folder (Empty Folder).

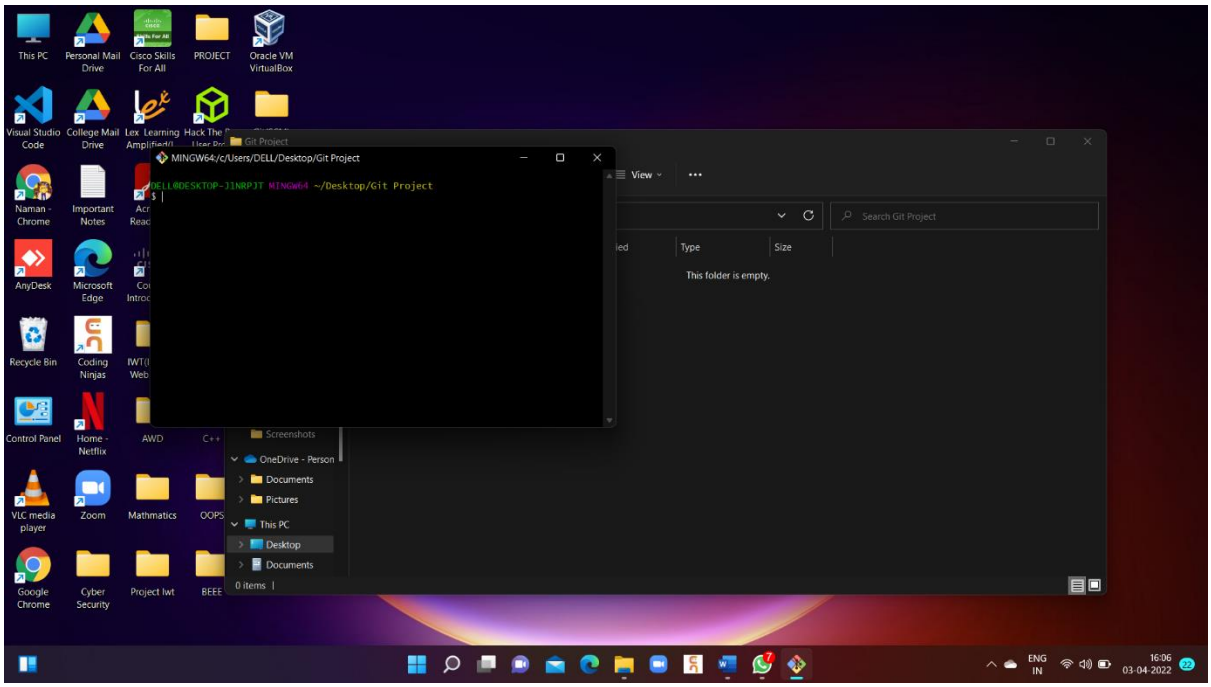


2. Go to location and Type Git Bash.lnk (It will AutoComplete it after writing Git) and Click on it.





3. It will open Git Bash Terminal Or Git Bash CLI(Command Line Interface).



4. Now We have to write all the Commands in this CLI.

5. On CLI Type “Git init.”

Git init→The git init command **creates a new Git repository**. It can be used to convert an existing, unversioned project to a Git repository or initialize a new, empty repository.

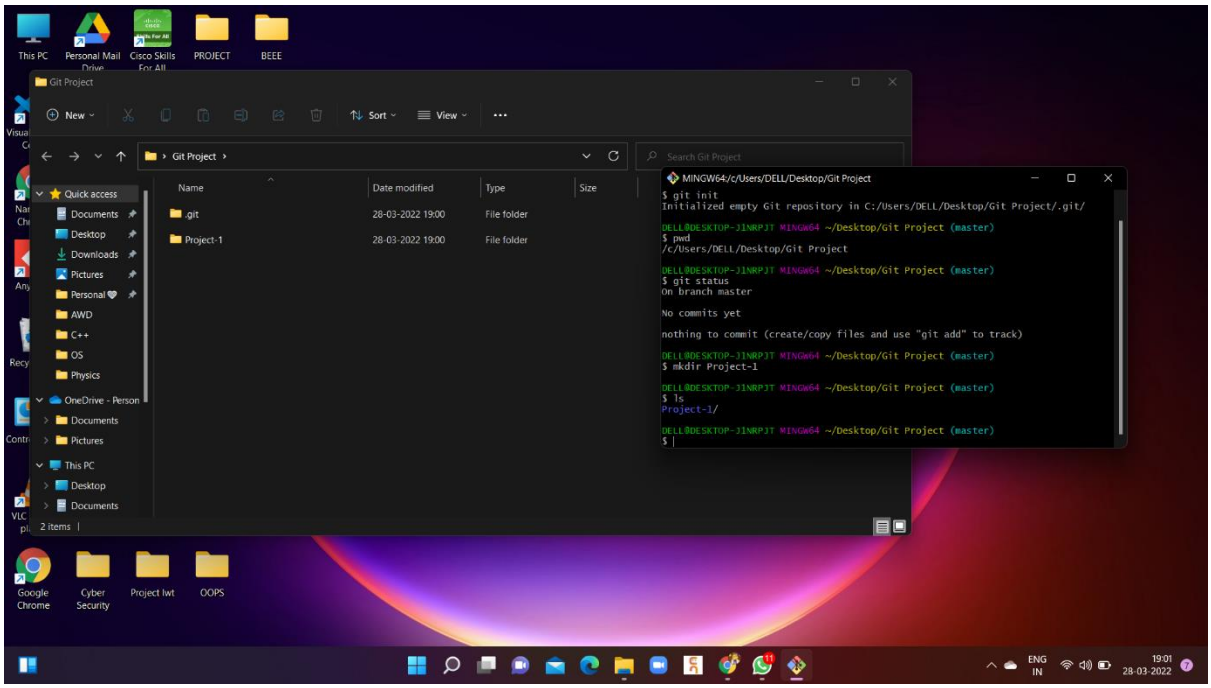
6. Type “pwd” , “git status” , “mkdir Project-1” and “ls”.

Pwd(Present Work Directory)→ The pwd command writes the full pathname of the current working directory to the standard output

Git status→ The git status command **displays the state of the working directory and the staging area**

Mkdir <Directory Name>→ The mkdir command in CLI allows users to **create or make new directories**.

ls→ The ls command is used **to list files or directories in Linux and other Unix-based operating systems**.



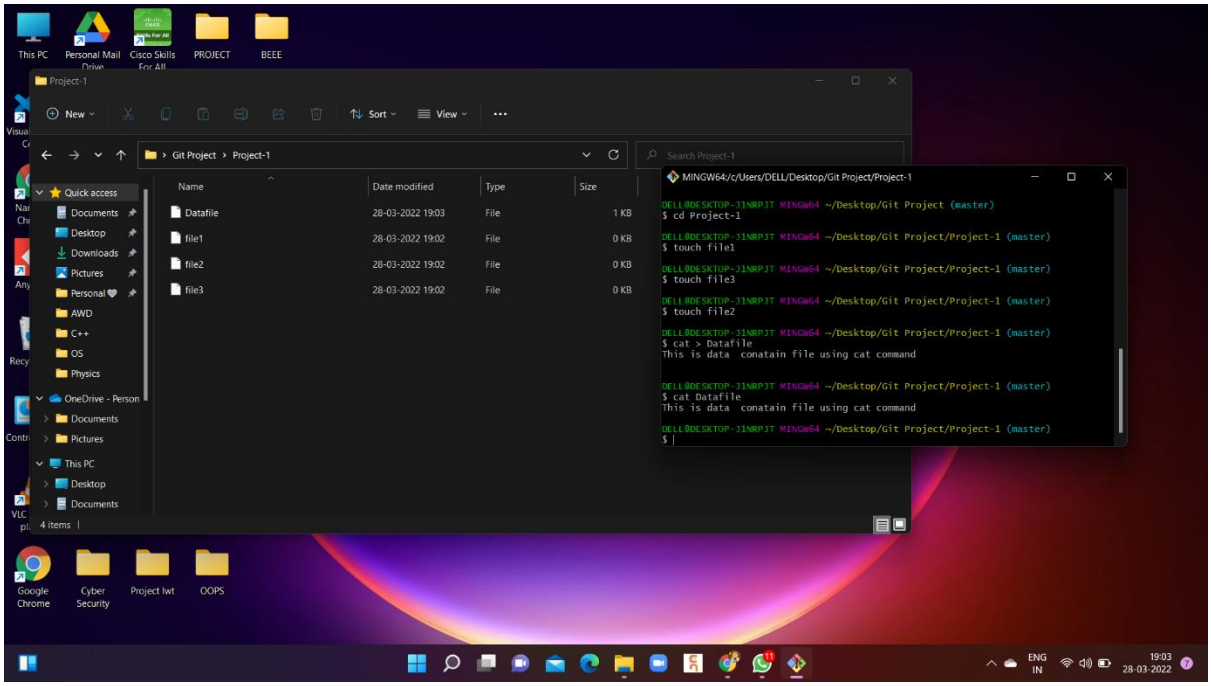
7. Type “cd Project-1” , “touch file1” , “touch file2” , “touch file3” ,“cat > Datafile” and “cat DataFile” .

Cd <Directory Name>→ The **cd command**, also known as chdir (change directory), is a command-line shell command used to change the current working directory in various operating system.

Touch <Filename>→ The **touch command** is a standard command which is used to create, change and modify timestamps of a file.

Cat <(Filename)→ **You can create new files and add content to them using the `cat` command.**

Cat (Filename)→ **It can be used to display the content of a file**

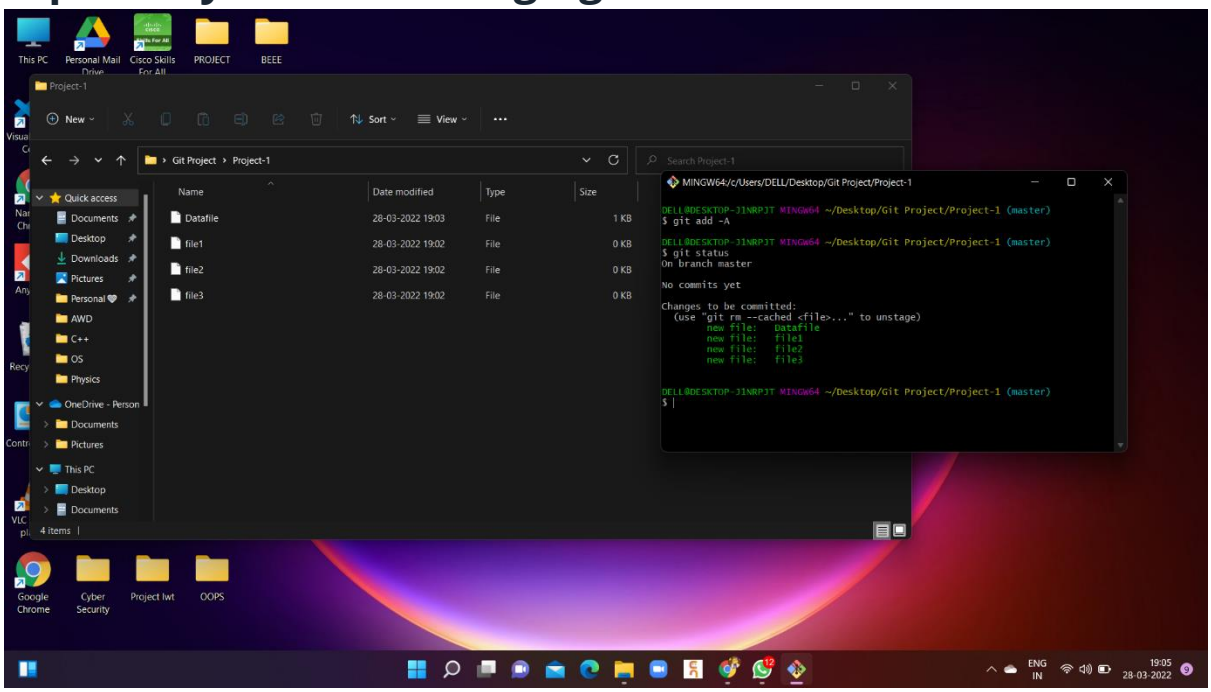


8. Type “git add -A” or git add <filename> (git add file1) and Check status using “git status” Command.

Git add <filename>→**The git add command adds a file to the Git staging area. This area contains a list of all the files you have recently changed**

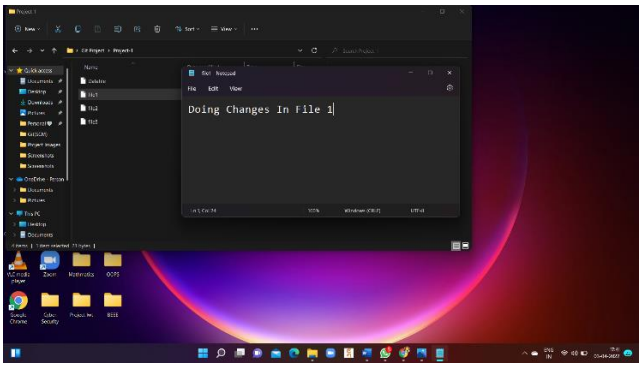
Git add -A→ **This command adds every change we made to files and folders from our**

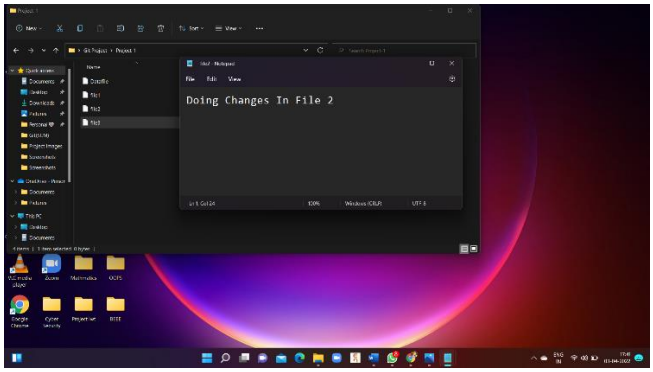
repository to the Git staging area.

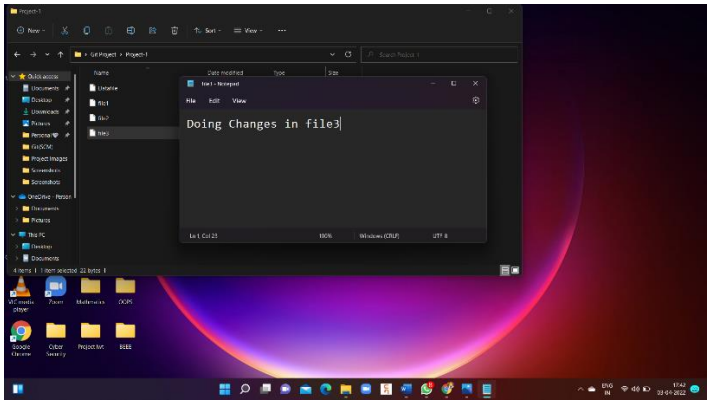


After Adding the files in Stagging area. Our Files are now ready to commit.

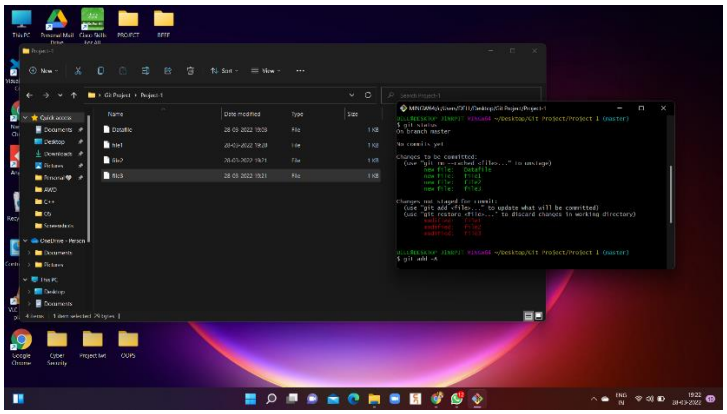
9. Open the Files in any editor like: “Notepad” and “VS code” and Do changes in it. And Save it.







Now Check the Status by using Command “git status” in CLI.



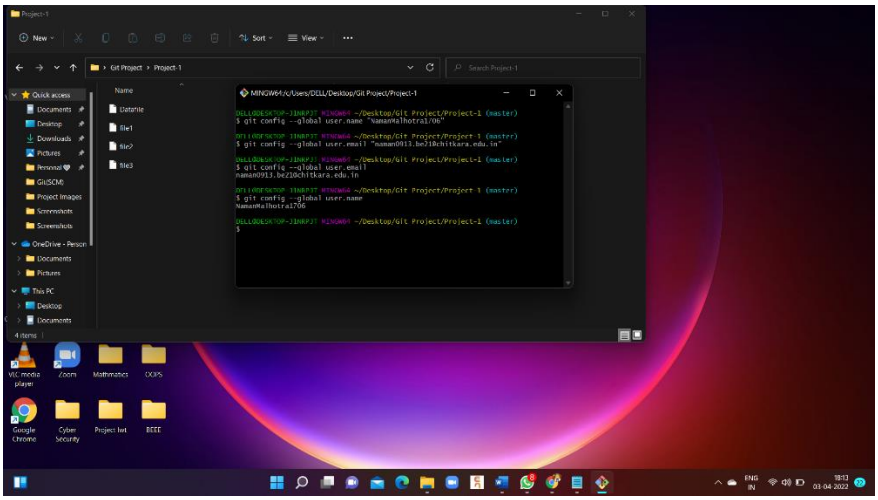
The Files are modified , We need to add these modified files in staging area again to do Commits.

10. Before Commit we need to give our username and email. This setup only needs to be done once on your computer.

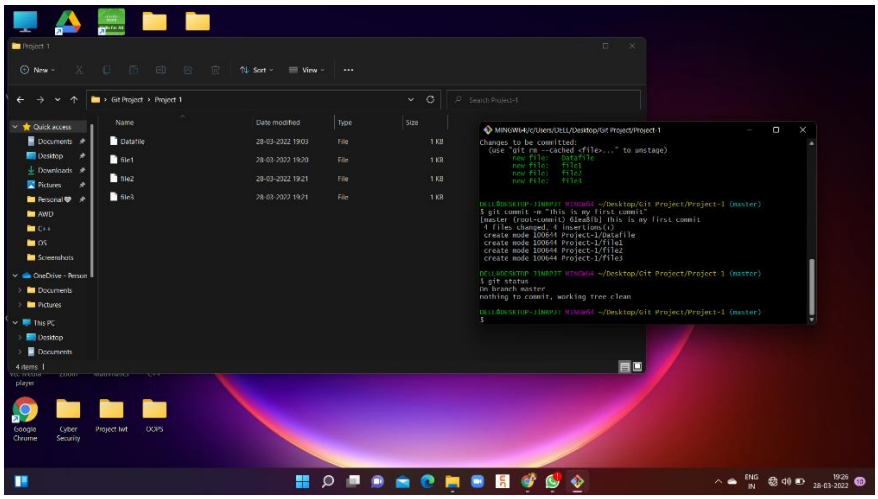
And this Can be done by using Command :

For UserName :git –config –global user.name”User Name”.

For UserEmail :git –config –global user.email”UserEmail”.

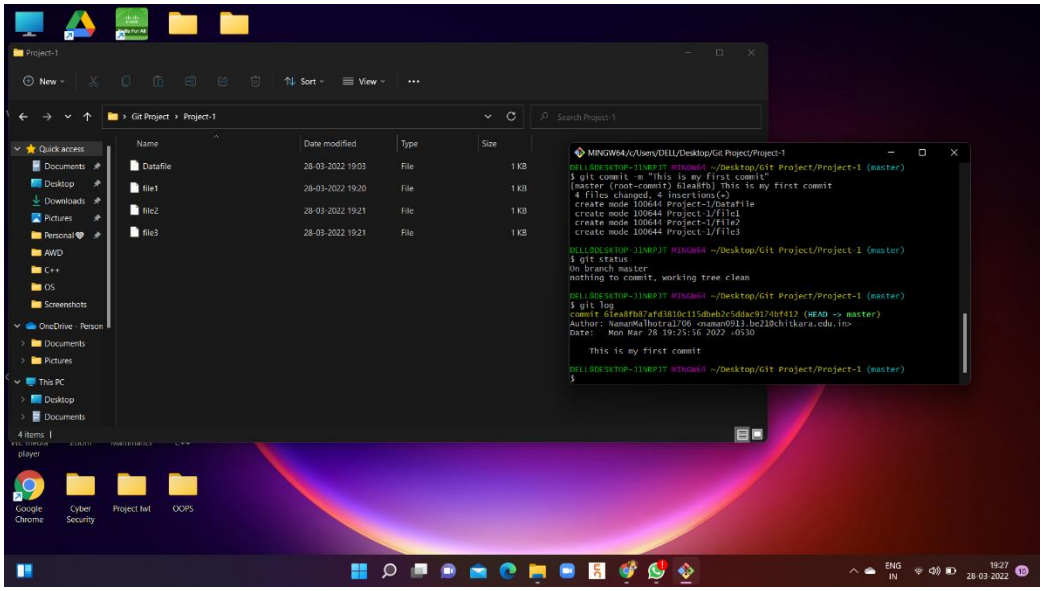


10. Type Command “git add -A” in CLI to add files in staging area. And Type “git commit -m “This is my first Commit””.



Git commit -m <"The Message to commit">→ The git commit command captures a snapshot of the project's currently staged changes. Committed snapshots can be thought of as "safe" versions of a project—Git will never change them unless you explicitly ask it to

11. Now Check Status and type Command "git log" in CLI.



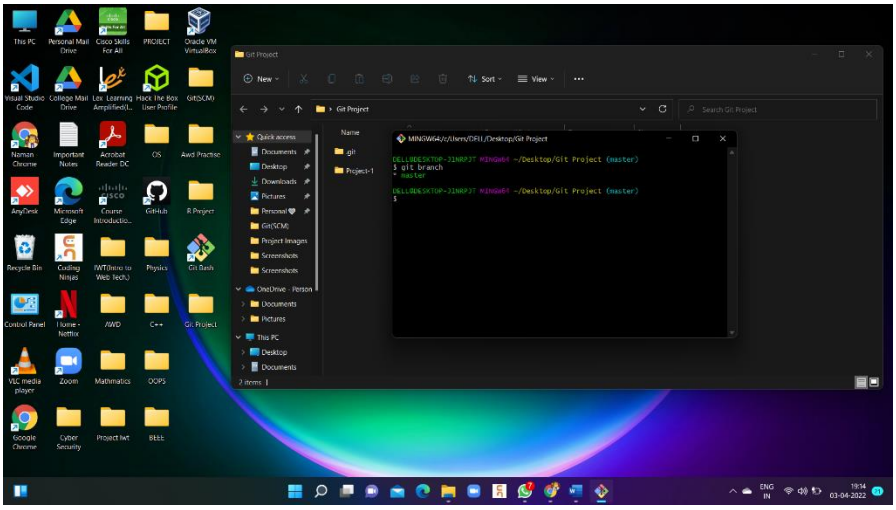
Git log→Git log is **a utility tool to review and read a history of everything that happens to a repository**

CHITKARA UNIVERSITY

Create and Visualize Branches

A branch is a version of the repository that diverges from the main working project. It is a feature available in most modern version control systems. A Git project can have more than one branch. These branches are a pointer to a snapshot of your changes. When you want to add a new feature or fix a bug, you spawn a new branch to summarize your changes.

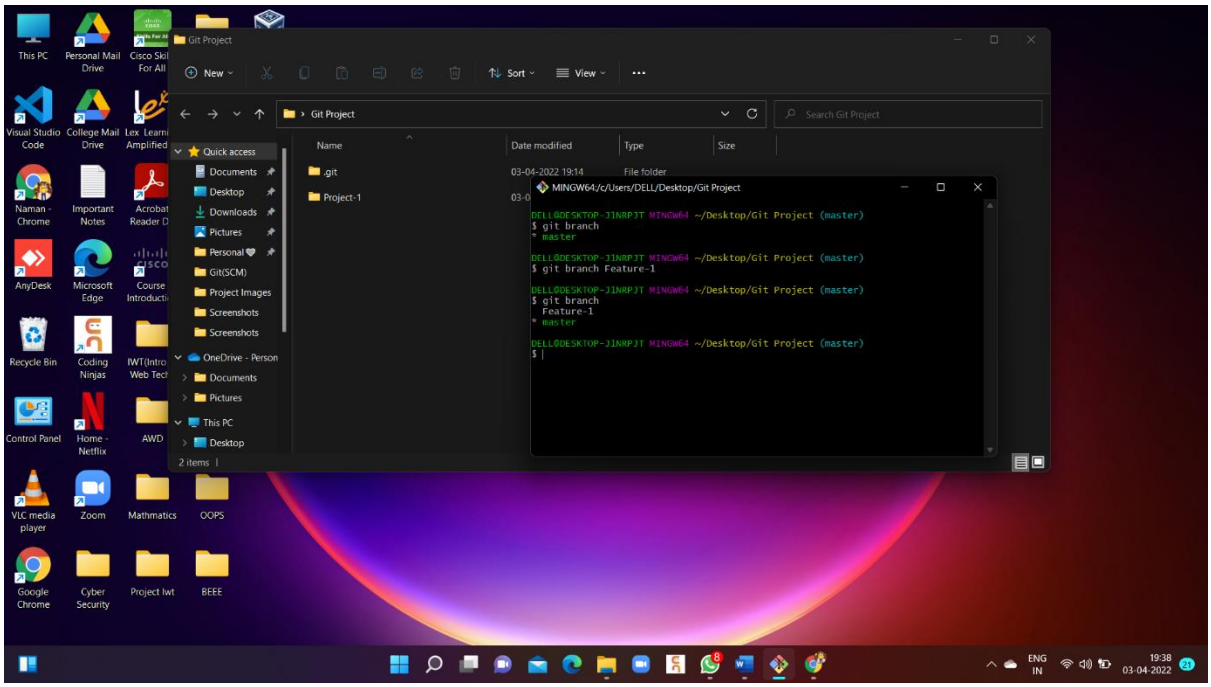
The master branch is a default branch in Git



Use “**git branch**” Command, gives us List of all branches in our Repo. If the branch name is in Green Colour It show, in which branch we are.

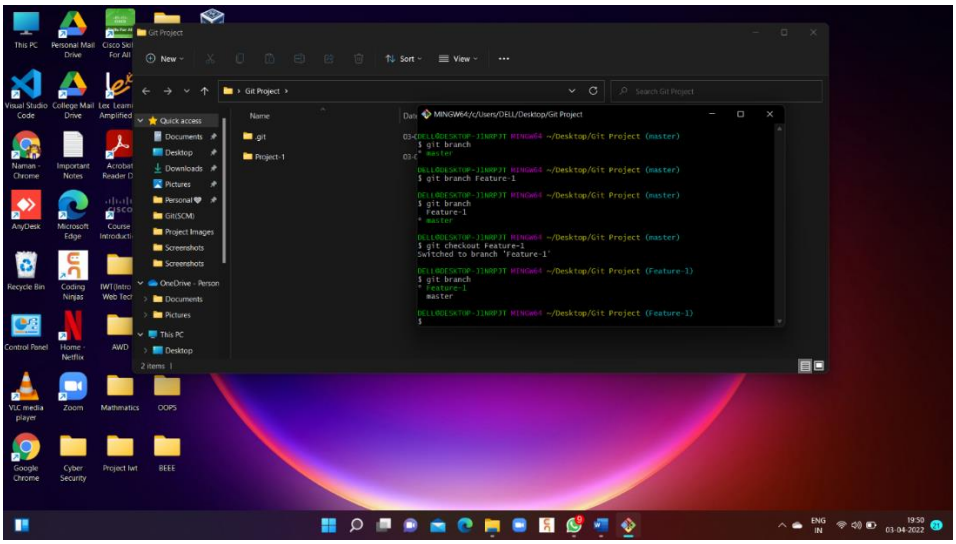
1. Write command “git branch Feature-1" on CLI

Git branch <branch name>→This Command Allow us to make a new Branch (which is Copy of Master Branch)

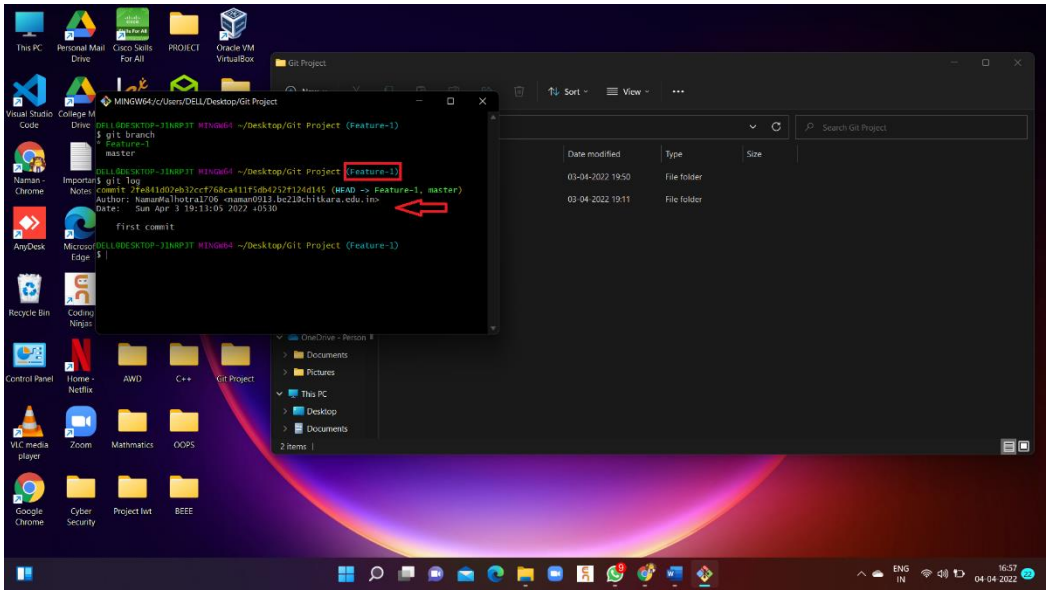


2. Type “git checkout Feature-1”.

git Checkout <branch name>→ This command allow us to Switch to another Branch (i.e) Feature-1.

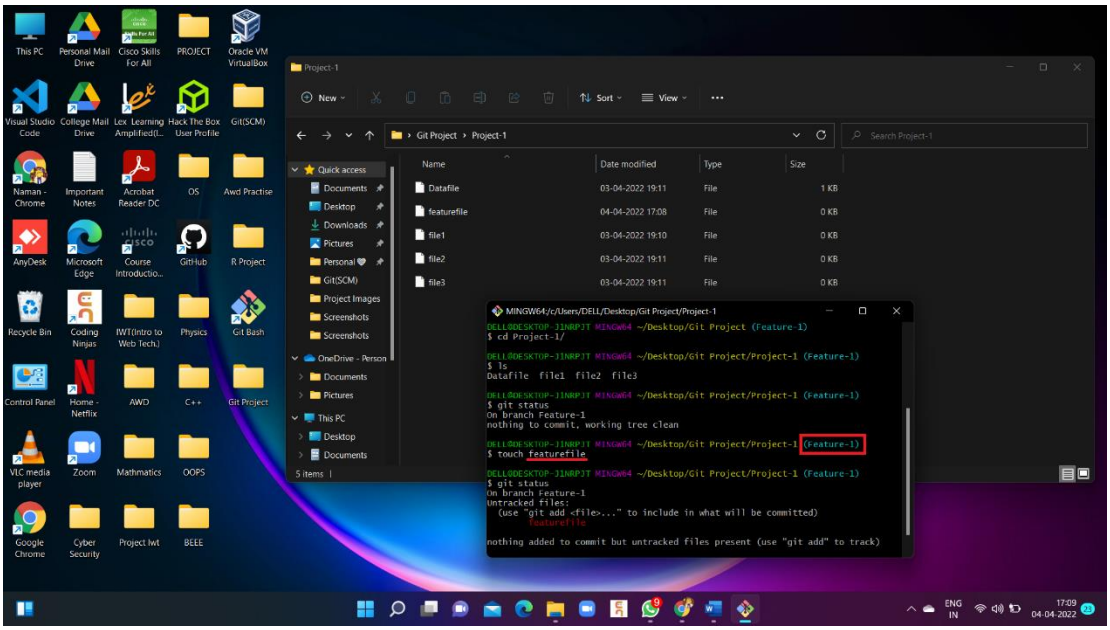


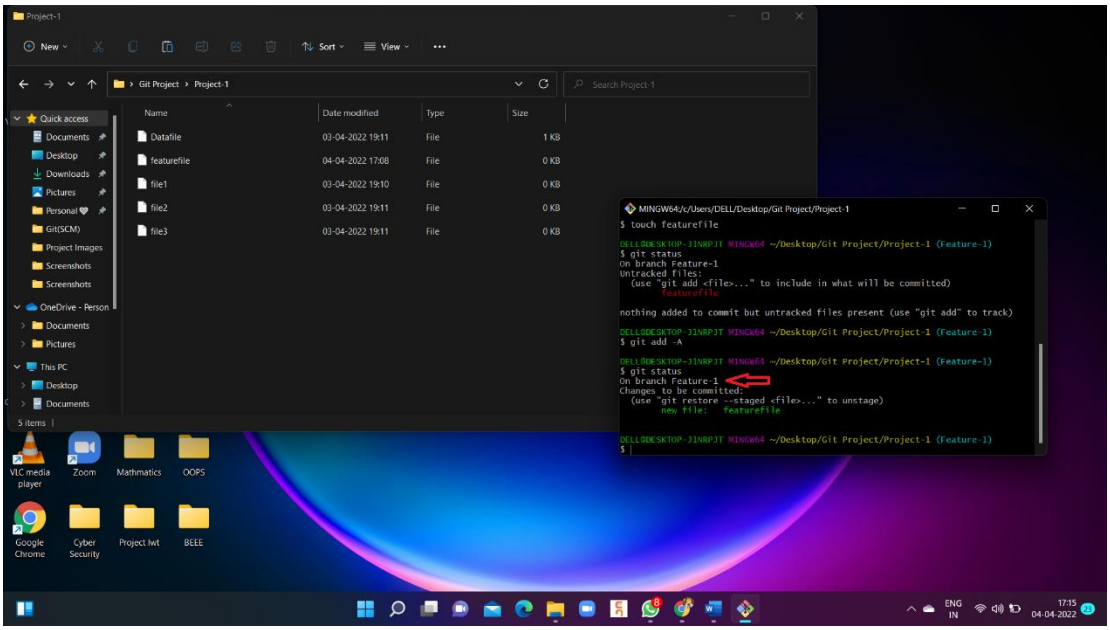
The Feature-1 branch is the copy of Master branch. It contains all the Commits done in master branch.



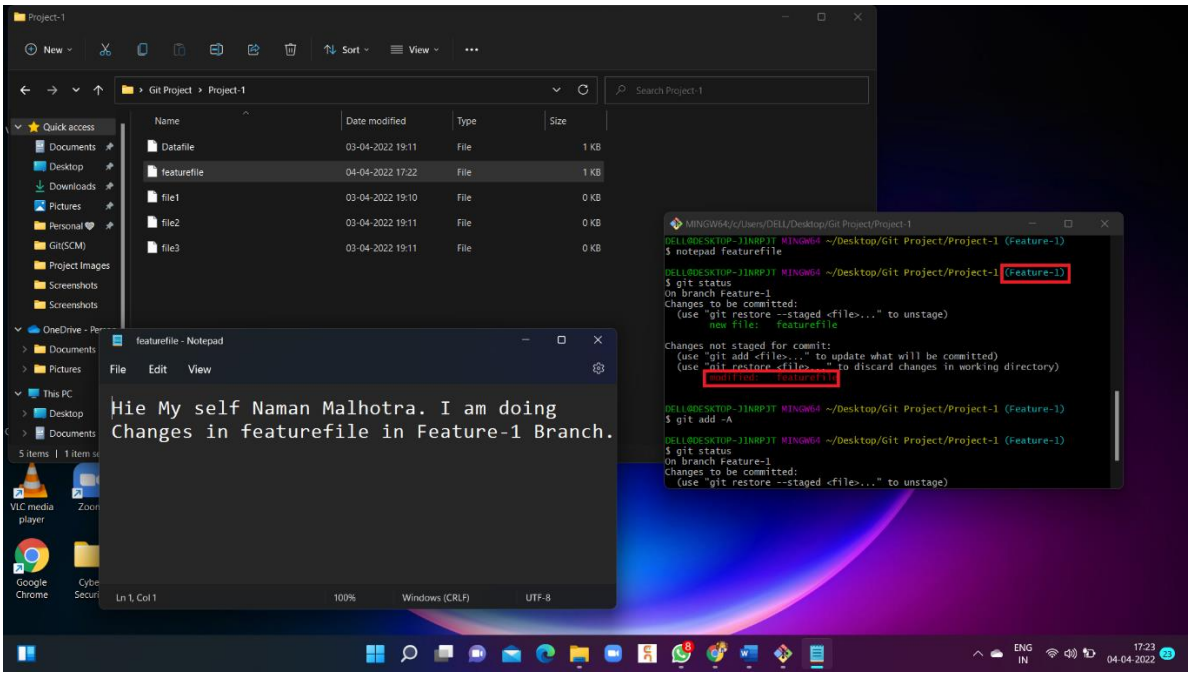
If we do changes in the Feature-1 branch it will no affect our master branch.

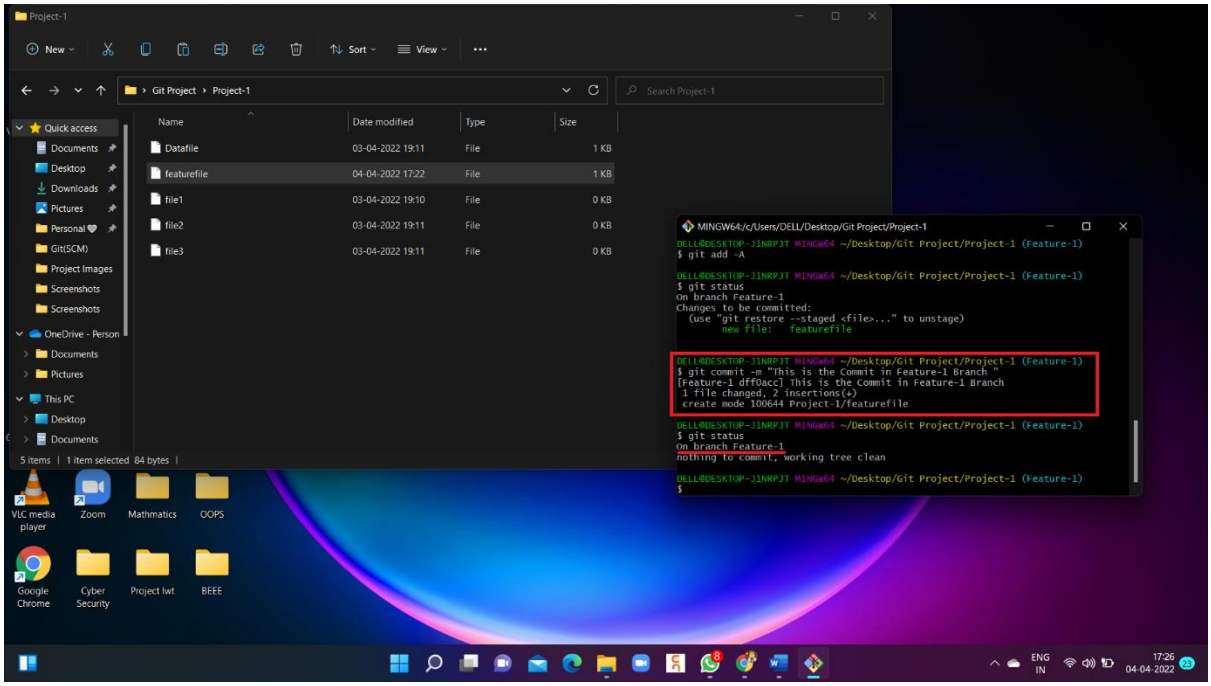
3.Creating a new file “featurefile” in Feature-1 branch and add it in staged area.



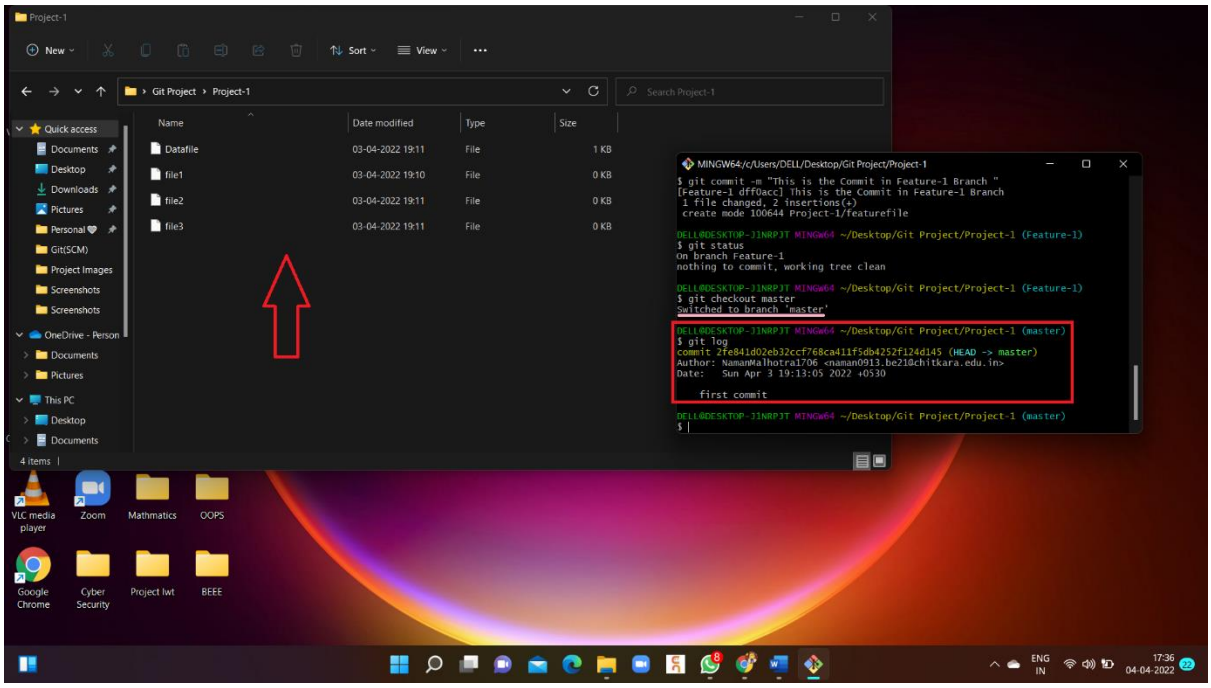


4. Do changes in the “featurefile” and commit it.





4. Now Switch to Master branch.



We Can see That After Switching to master branch it doesn’t show “featurefile” and Commit done in Feature-1 branch.

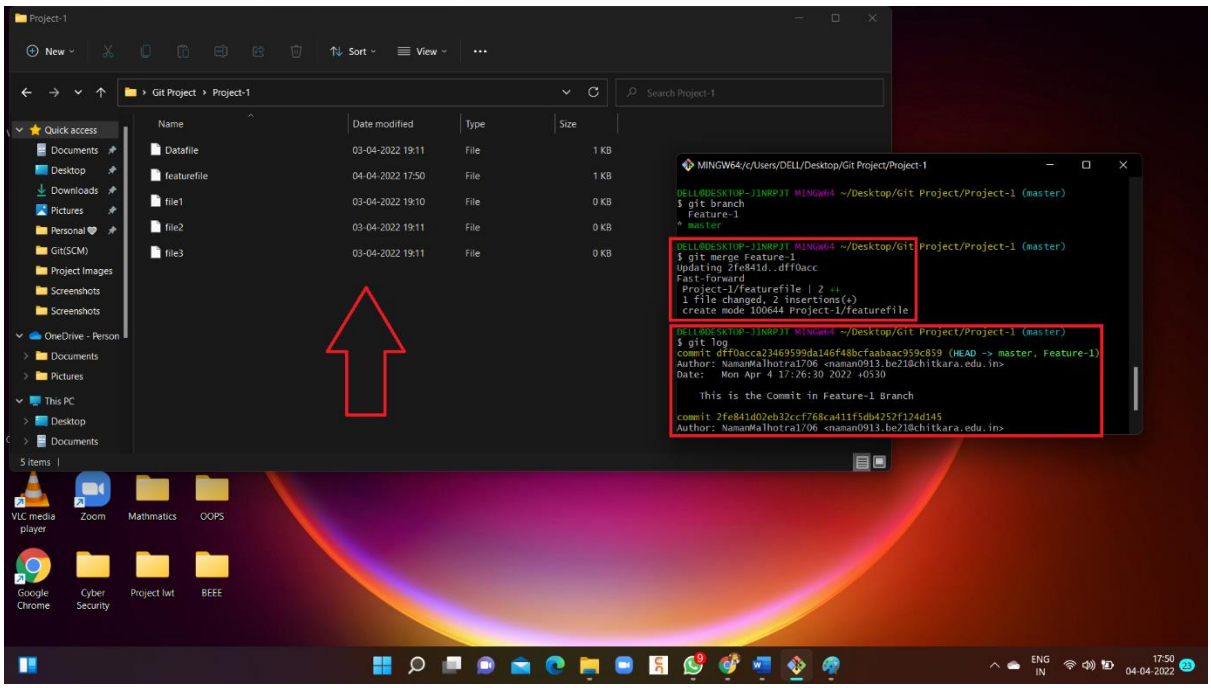
Its Means Newly Created branch Doesn’t have impact on Master branch. If we do changes in the branch it will not affect our data in the master branch.

If the change or added data in New branch is correct and you have to merge that data in master branch then,

5. On CLI, First switch to master branch.

6. Type Command “**git merge Feature-1”.**

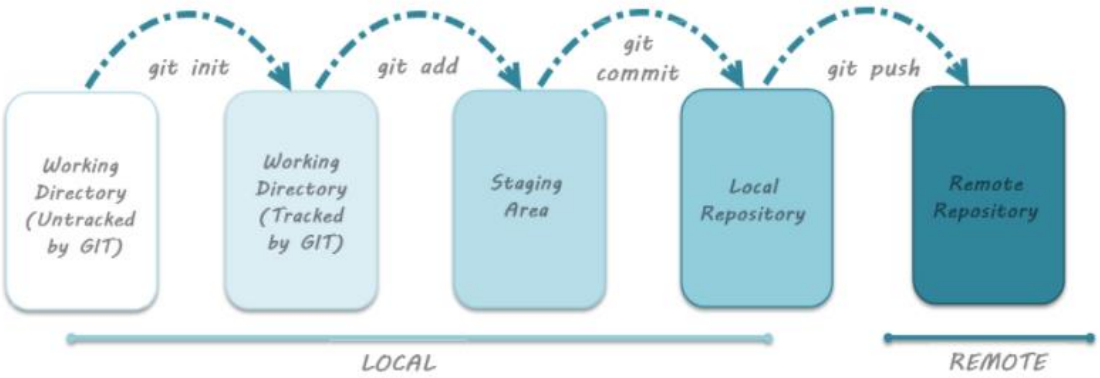
Git merge <branch Name>→ it merge the branch into another branch.



CHITKARA UNIVERSITY

The changes or added data merge in master branch.

Git Lifecycle Description



Files in a ***Git*** project have various stages like ***Creation, Modification, Refactoring***, and ***Deletion*** and so on. Irrespective of whether this project is tracked by Git or not, these phases are still prevalent. However when a project is under Git version control system, they are present in three major Git states in addition to these basic ones. Here are the three Git states:

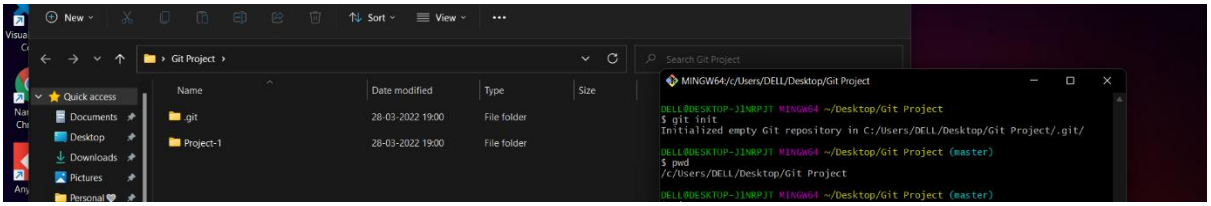
- *Working directory*
- *Staging area*
- *Git directory*

These stages are the essence of Git. You get great flexibility in tracking the files due to these stages that files can reside in under Git. Let's understand each of these states one by one.

Working Directory

Consider a project residing in your local system. This project may or may not be tracked by Git. In either case, this project directory is called your Working directory.

Working directory is the directory containing hidden .git folder.



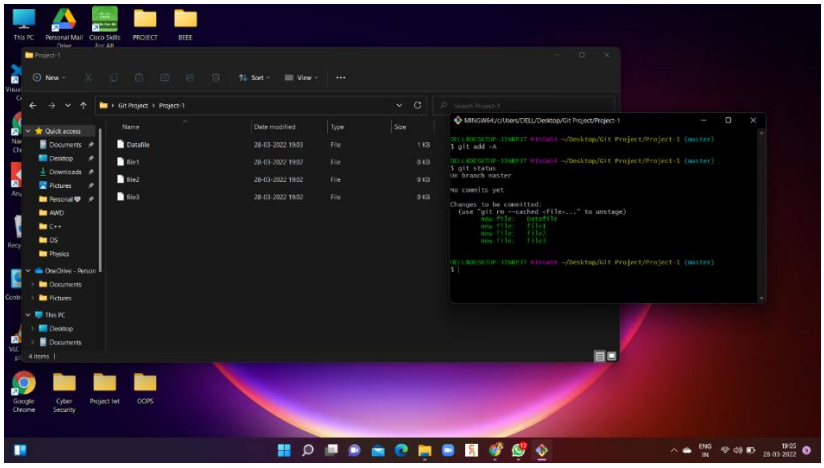
Note: ***git init*** - Command to initialize a Git repository

Let's assume that this directory is now tracked by Git. That is we've created a Git repository in this existing project directory. So, a ***hidden .git folder*** is initialized therein.
In this state, Git is just aware of the files in the project. It doesn't track the files yet. .git is the main repository and it keeps all the project history . It can be used to ***roll back***, ***can fetch the version, saves the snapshots***. Its most important feature is that in it almost ***every operation is local*** i.e. no internet is needed. GIT has integrity and maintains it.

To track the files, we've to commit these files by first adding the files to the staging area. This brings us to the next state in Git life-cycle.

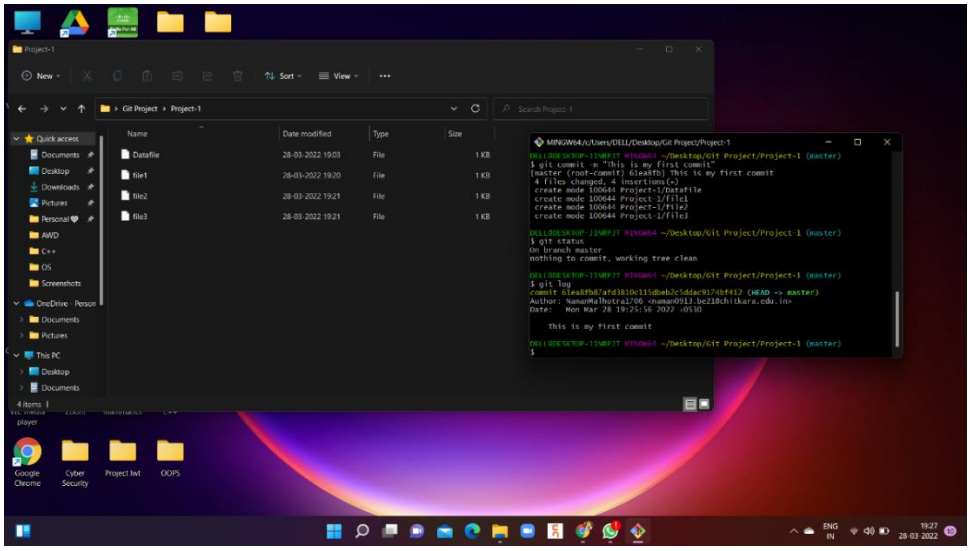
Staging Area

Now, to track the different versions of our files we use the command **git add**. We can term a staging area as a place where different versions of our files are stored. git add command copies the version of your file from your working directory to the staging area. We can, however, choose which files we need to add to the staging area because in our working directory there are some files that we don't want to get tracked, examples include *node modules, env files, temporary files*, etc. Indexing in Git is the one that helps Git in understanding which files need to be added or sent



Git Directory

Now since we have all the files that are to be tracked and are ready in the staging area, we are ready to commit our files using the git commit command. Commit helps us in keeping the track of the metadata of the files in our staging area. We specify every commit with a message which tells what the commit is about. Git preserves the information or the metadata of the files that were committed in a Git Directory which helps Git in tracking files and basically it preserves the photocopy of the committed files. Commit also stores the name of the author who did the commit, files that are committed, and the date at which they are committed along with the commit message.



CHITKARA UNIVERSITY

SUMMARY

The tasks performed in all are listed below:

1. Setting up of Git Client,
2. Setting up GitHub Account,
3. Program to Generate logs
4. Create and visualize branches
5. Git lifecycle description