



CHITKARA

UNIVERSITY

Subject Name: **Source Code Management**

Subject Code: **CS181**

Cluster: **Beta**

Department: **DCSE**

Submitted By:

Abhishek

2110990060

G1

Submit to:

Mr. Monit Kapoor

LIST OF TASKS

S. No.	Task Title	Page No.
1.	Setting up of Git Client	2 – 12
2.	Setting up GitHub Account	13 - 16
3.	Generate logs	17 - 24
4.	Create and Visualize Branches	25 - 28
5.	Git lifecycle description	29 - 31

1. Setting Up Git Client

What is Git?

Git is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency.

Why we need Git?

Git tracks the changes you make to files, so you have a record of what has been done, and you can revert to specific versions should you ever need to. Git also makes collaboration easier, allowing changes by multiple people to all be merged into one source.

Installing Git on Windows

Download and install the latest version of Git for Windows.

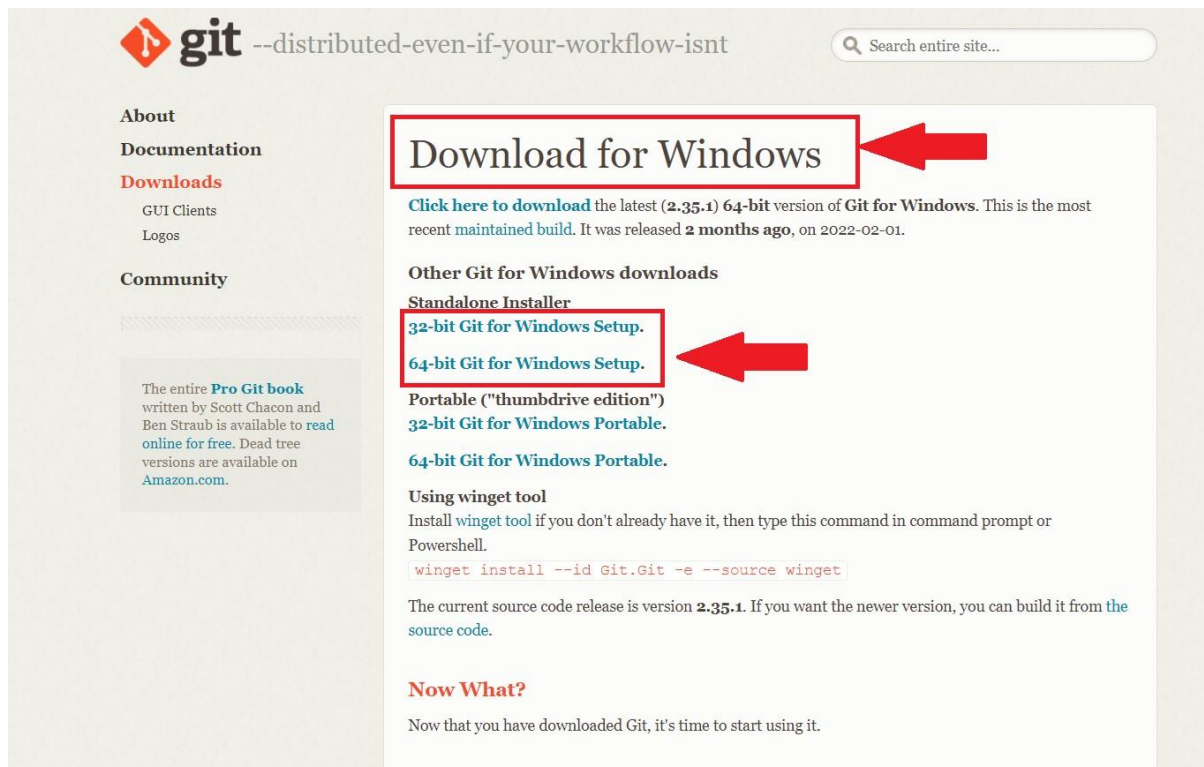
About 1,15,00,00,000 results (0.42 seconds)

<https://git-scm.com> › download › win**Downloading Package - Git** ✓

Download for **Windows**. Click here to download the latest (2.35.1) 32-bit version of **Git** for **Windows**. This is the most recent maintained build.

<https://git-scm.com> › downloads**Downloads - Git** ✓

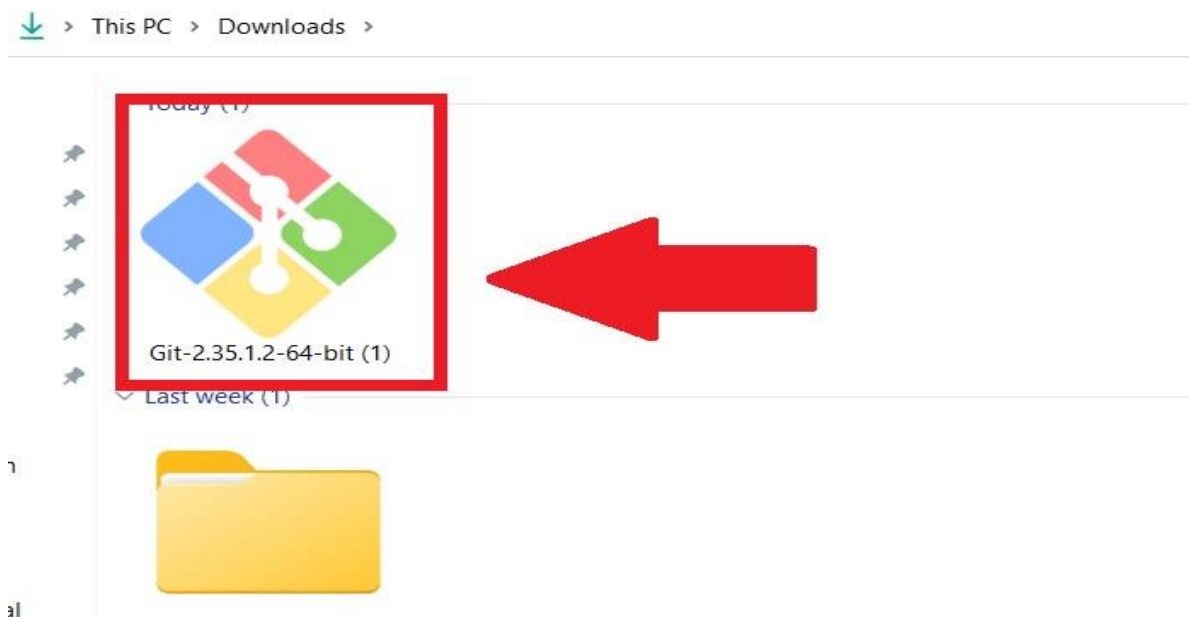
Downloads. macOS · **Windows** · Linux/Unix · Older releases are available and the **Git** source repository is on GitHub.



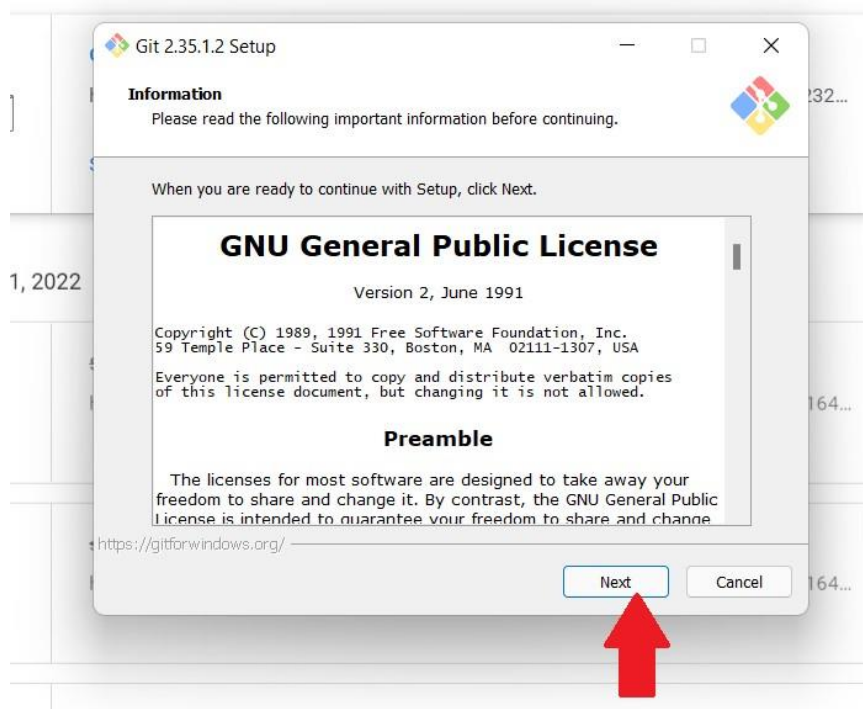
The screenshot shows the Git website with a sidebar on the left containing links for 'About', 'Documentation', 'Downloads' (highlighted), 'GUI Clients', 'Logos', and 'Community'. The main content area is titled 'Download for Windows' (highlighted with a red box and arrow). Below this title, it says 'Click here to download the latest (2.35.1) 64-bit version of Git for Windows. This is the most recent maintained build. It was released 2 months ago, on 2022-02-01.' Under the heading 'Other Git for Windows downloads', there are three sub-sections: 'Standalone Installer' (with '32-bit Git for Windows Setup.' and '64-bit Git for Windows Setup.' highlighted by red boxes and arrows), 'Portable ("thumbdrive edition")' (with '32-bit Git for Windows Portable.' and '64-bit Git for Windows Portable.' listed), and 'Using winget tool' (with instructions to install winget and a command: 'winget install --id Git.Git -e --source winget'). At the bottom, it says 'The current source code release is version 2.35.1. If you want the newer version, you can build it from the source code.' and 'Now What?' with the text 'Now that you have downloaded Git, it's time to start using it.'

Launch Git Installer

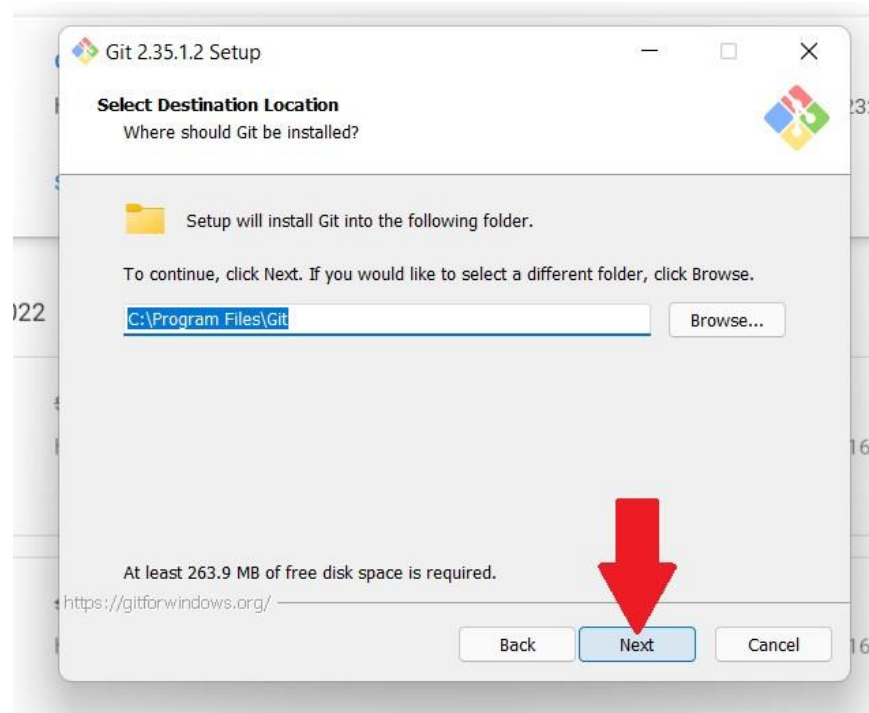
Go to **Downloads** in **This PC** and launch the installer.



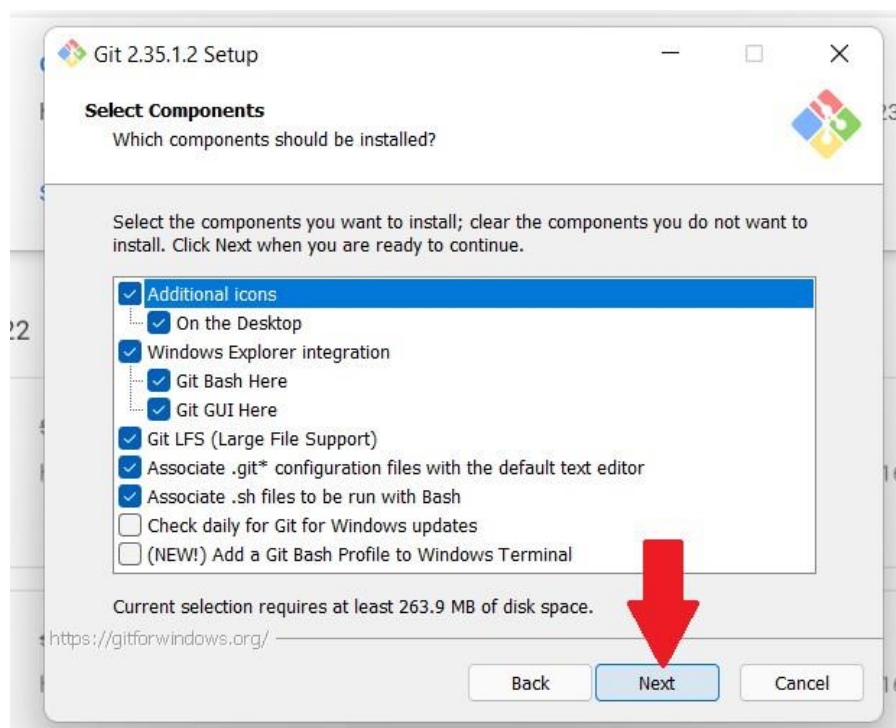
- Allow the app to make changes to your device by clicking Yes on the User Account Control dialog that opens.
- Review the GNU General Public License, and when you're ready to install, click Next.

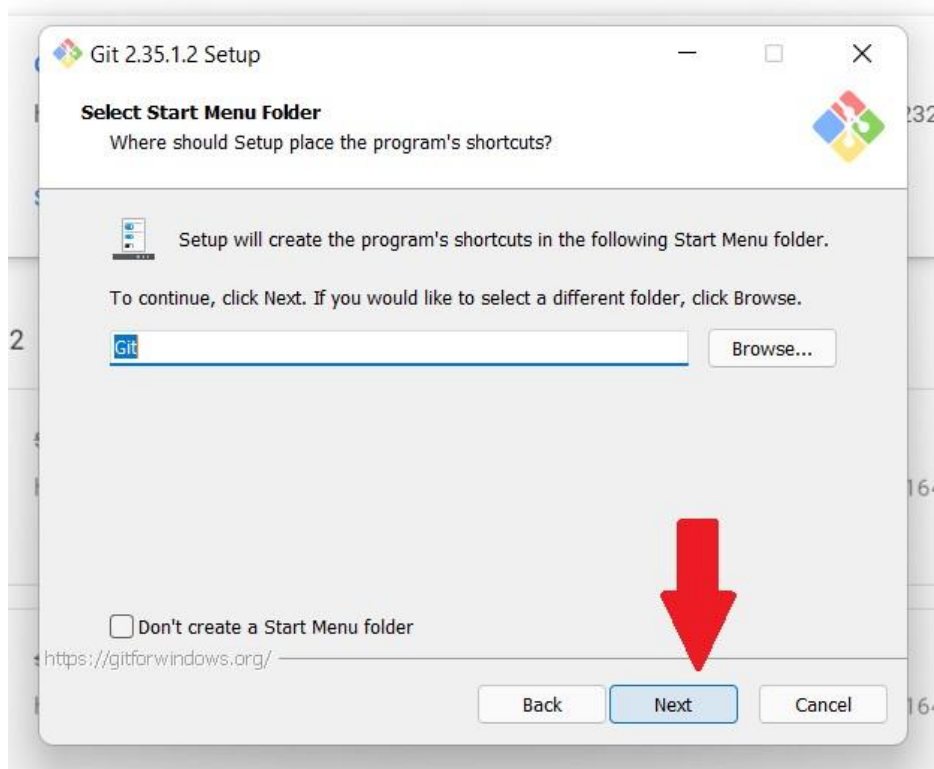


- Leave the default location unless you have reason to change it, and click **Next**.

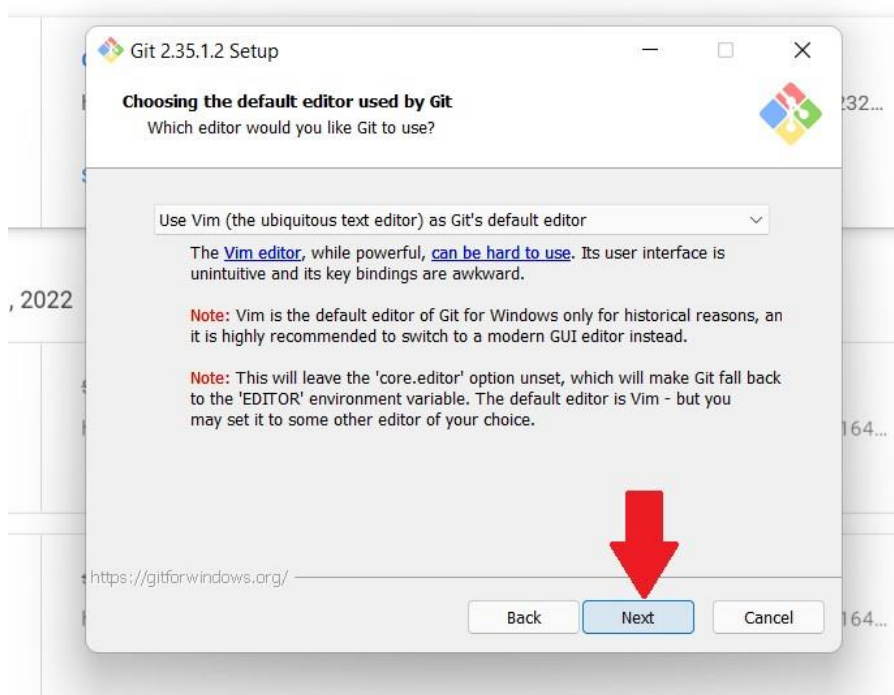


- For git bash shortcut at desktop, tick the box named **Additional icons**.

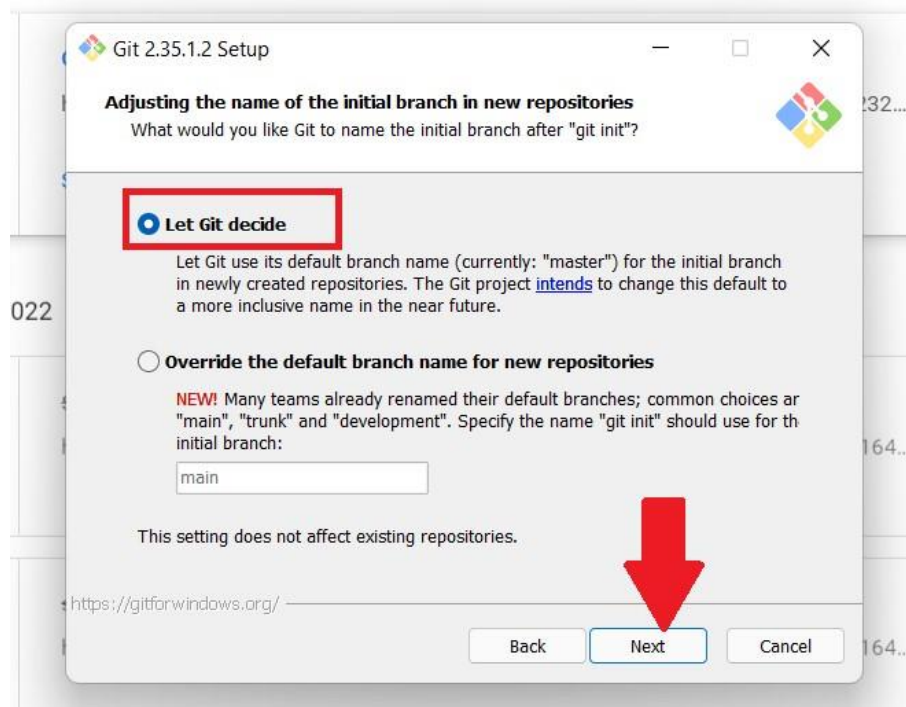




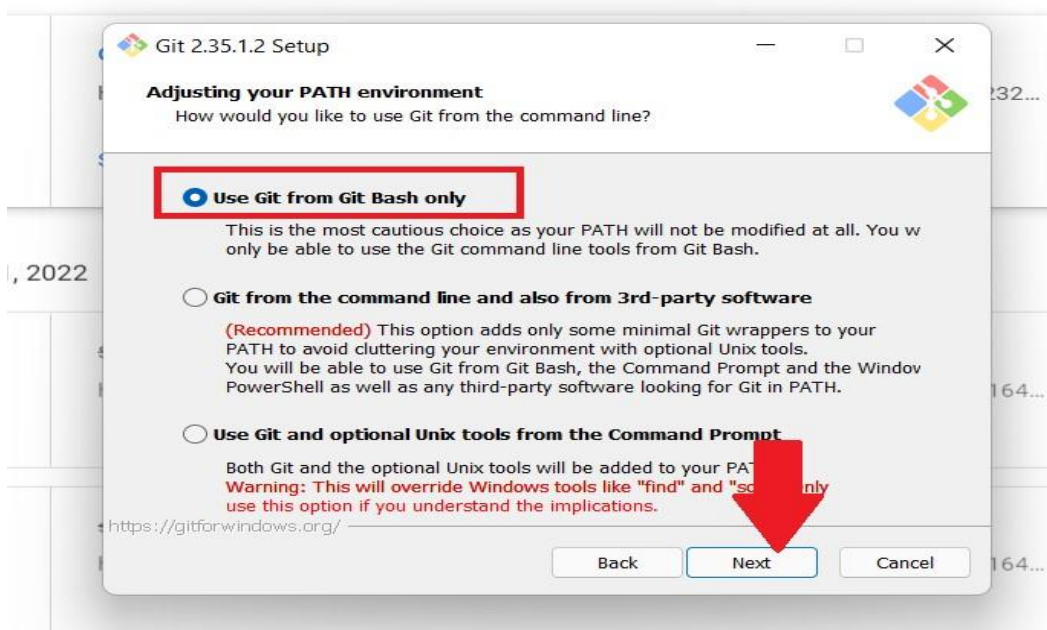
Select a text editor you'd like to use with Git. Use Vim (or whichever text editor you prefer) and click Next.



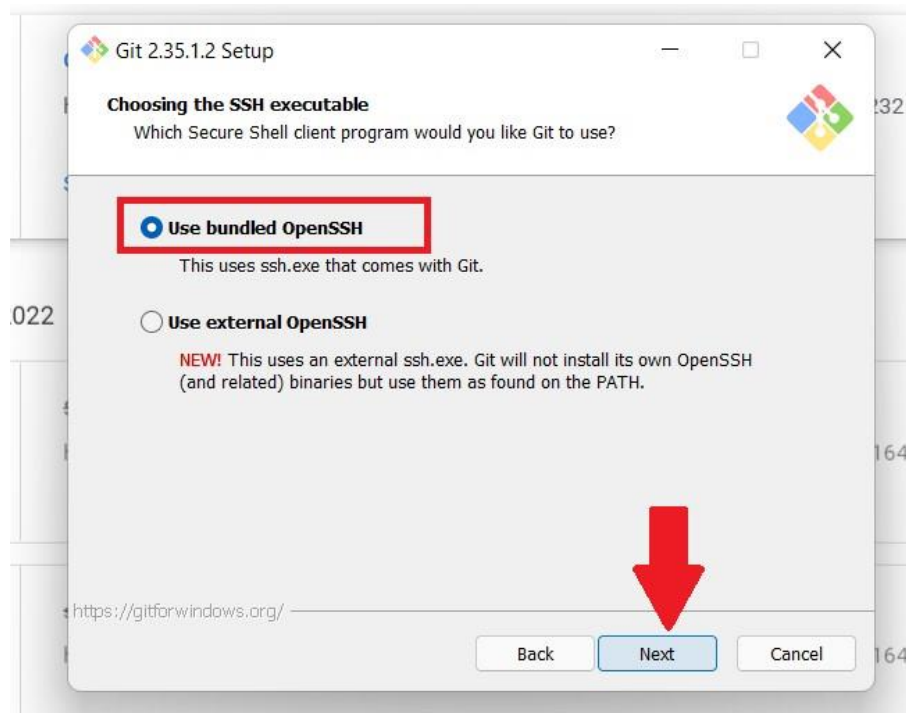
- The next step allows you to choose a different name for your initial branch. The default is 'master.' Unless you're working in a team that requires a different name, leave the default option and click Next.



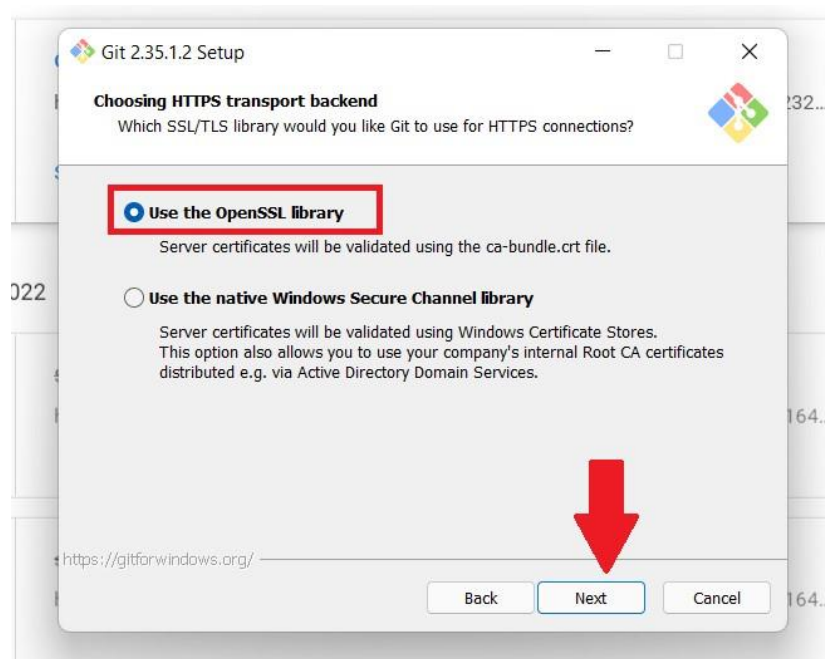
- This installation step allows you to change the PATH environment. The PATH is the default set of directories included when you run a command from the command line. Change it to Use Git from Git Bash. Click Next



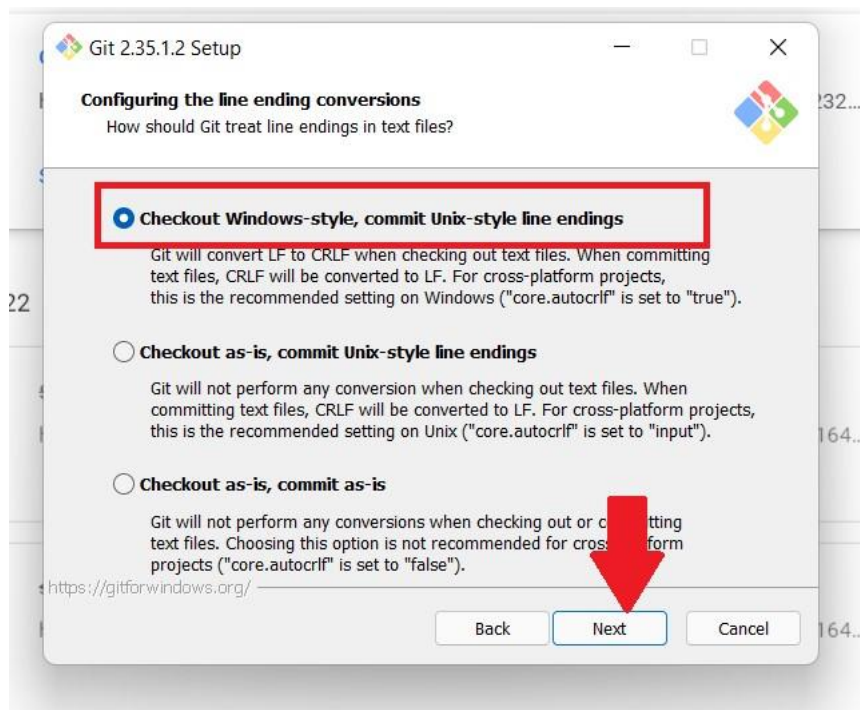
- The installer now asks which SSH client you want Git to use. Git already comes with its own SSH client, so if you don't need a specific one, leave the default option and click Next.



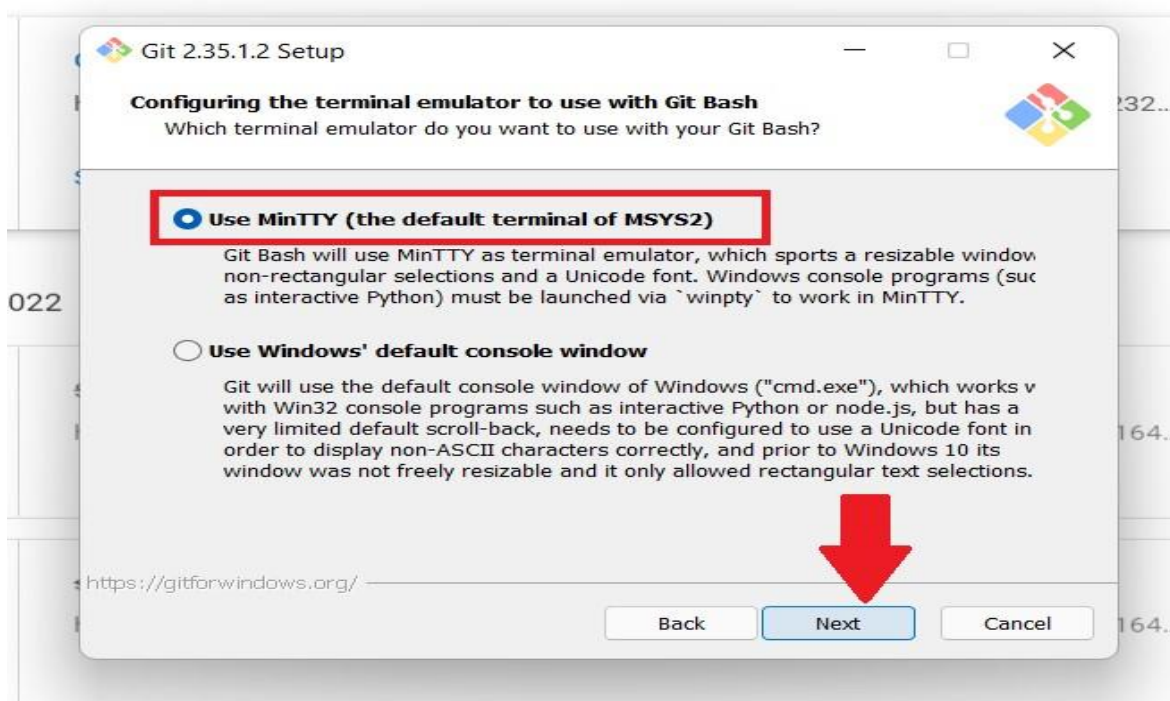
The next option relates to server certificates. Most users should use the default. If you're working in an Active Directory environment, you may need to switch to Windows Store certificates. Click Next.



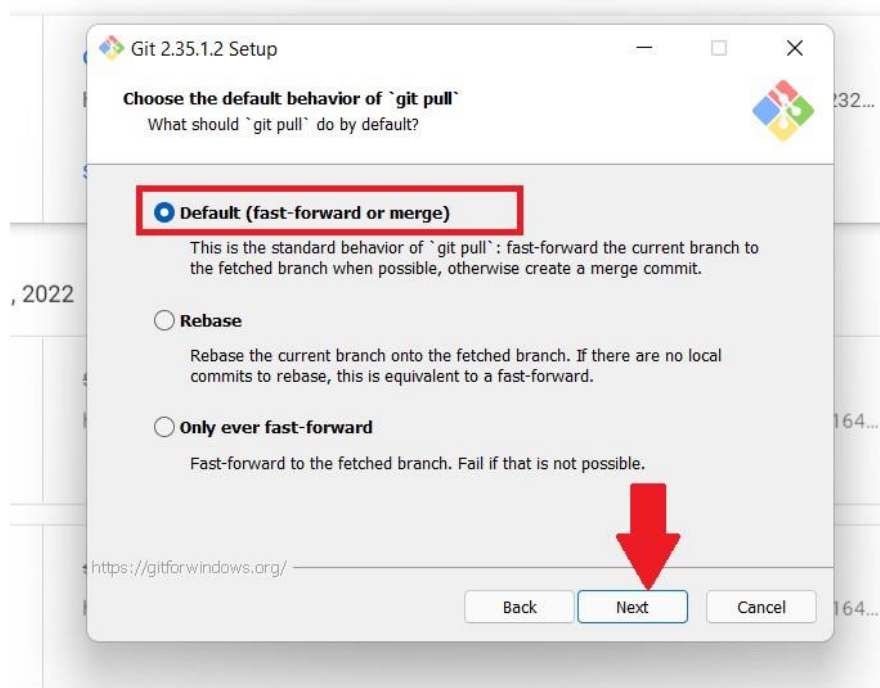
- The next selection converts line endings. It is recommended that you leave the default selection. This relates to the way data is formatted and changing this option may cause problems.



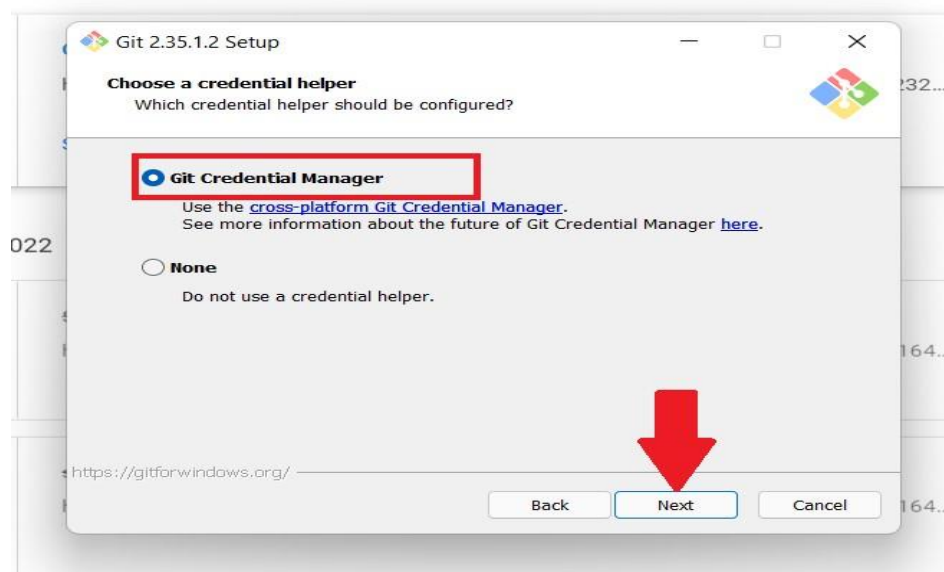
- Choose the terminal emulator you want to use. The default MinTTY is recommended, for its features. Click **Next**.



- 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.

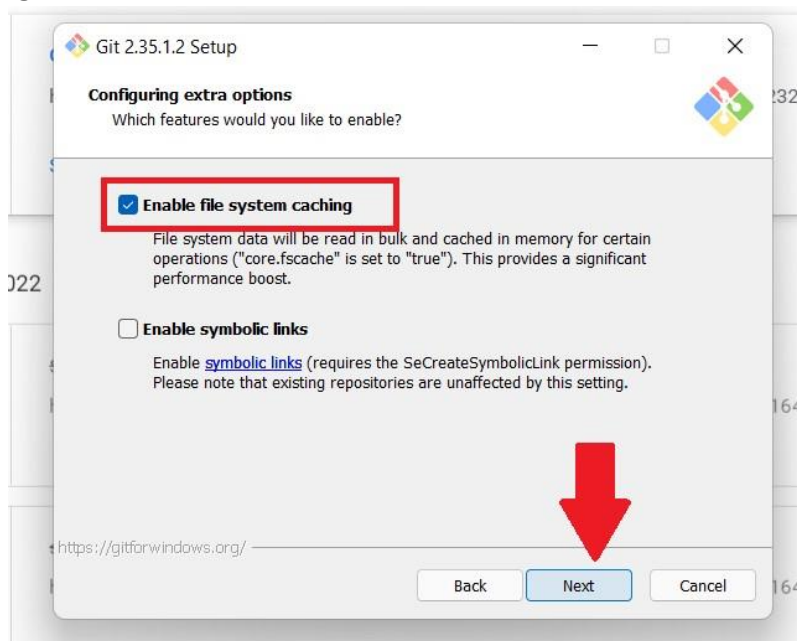


- Next you should choose which credential helper to use. Git uses credential helpers to fetch or save credentials. Leave the default option as it is the most stable one, and click **Next**.

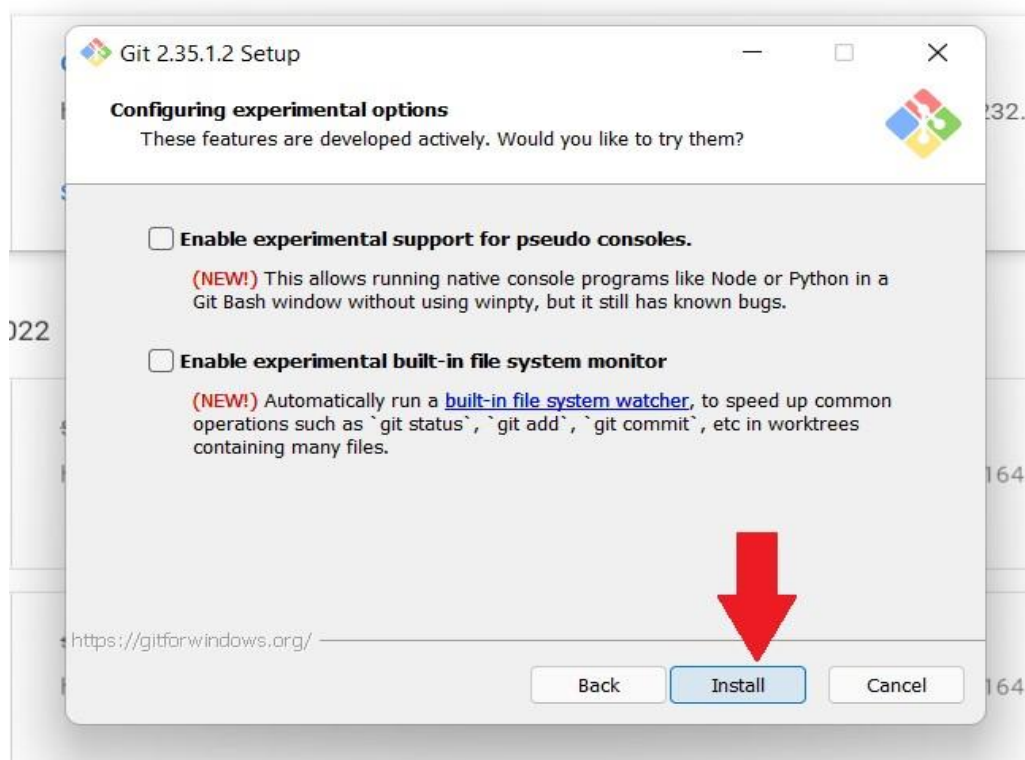


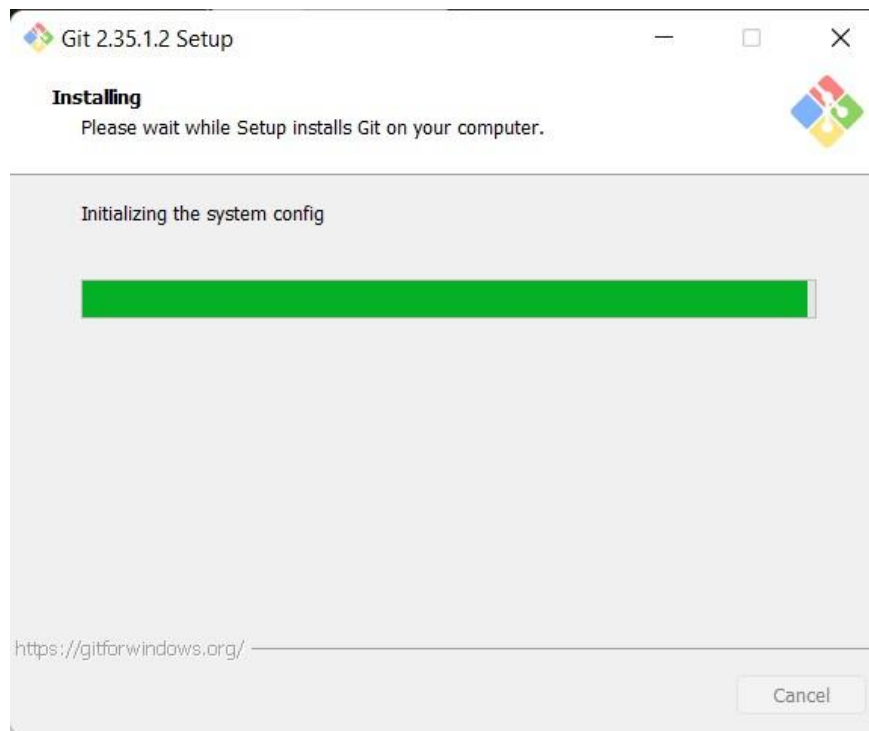
Additional Customization Options

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**.

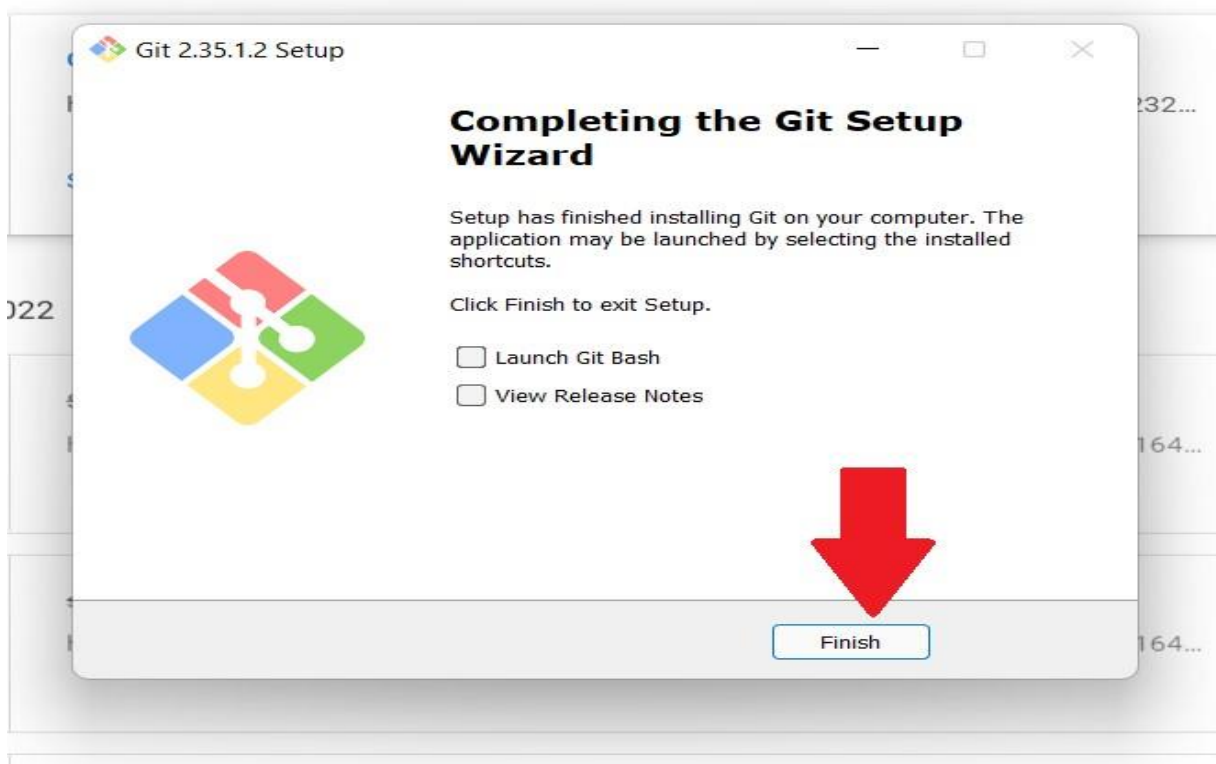


- Depending on the version of Git you're installing, it may offer to install experimental features. Unless you are feeling adventurous, leave them unchecked and click **Install**.





- Once the installation is complete, if you want to view the Release Notes or Launch Git Bash tick the box, then click Finish.



2. Setting Up GitHub Account

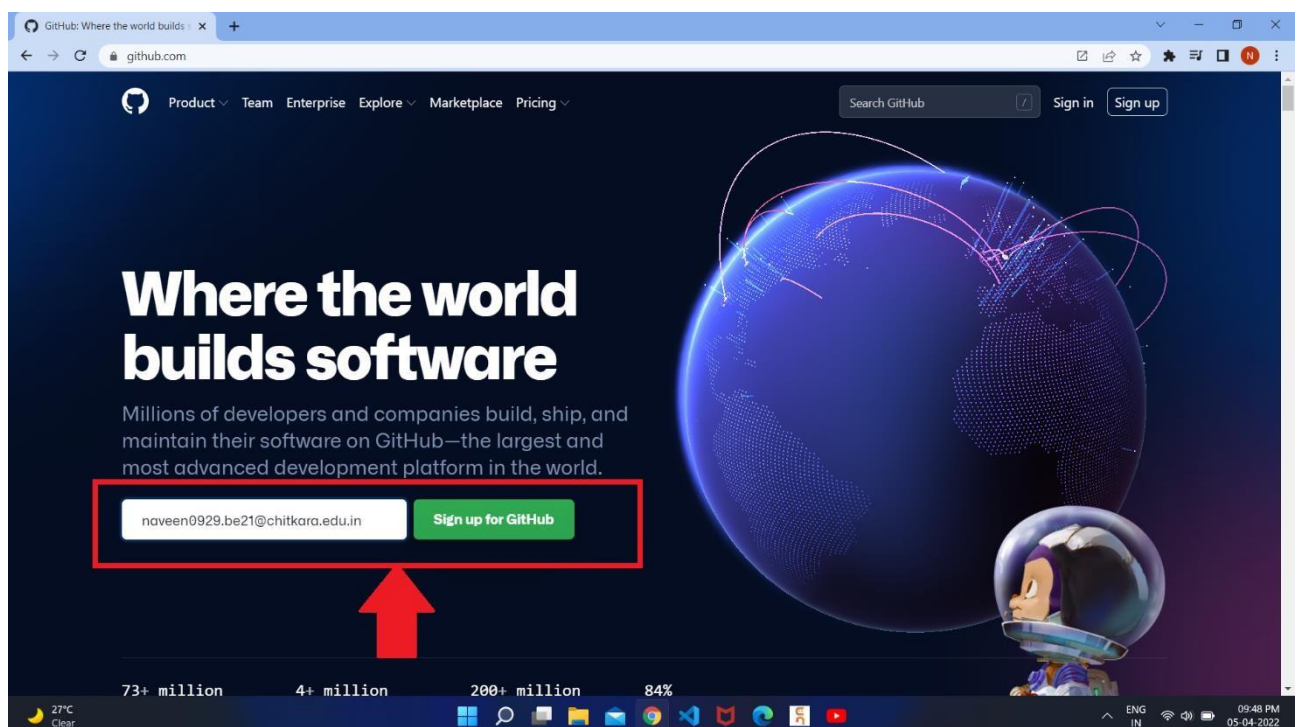
The first steps in starting with GitHub are to create an account, choose a product that fits your needs best, verify your email, set up two-factor authentication, and view your profile.

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.

2. Choosing your GitHub product

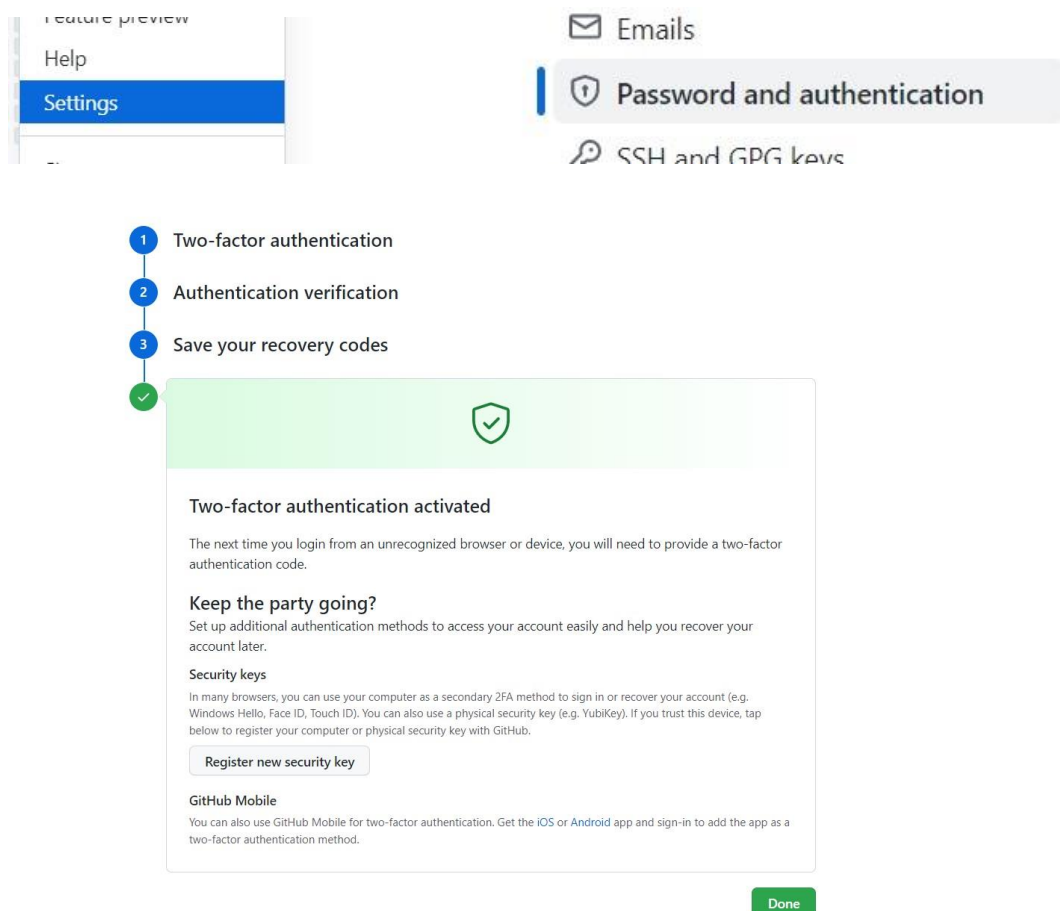
You can choose GitHub Free or GitHub Pro to get access to different features for your personal account. You can upgrade at any time if you are unsure at first which product you want.

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.

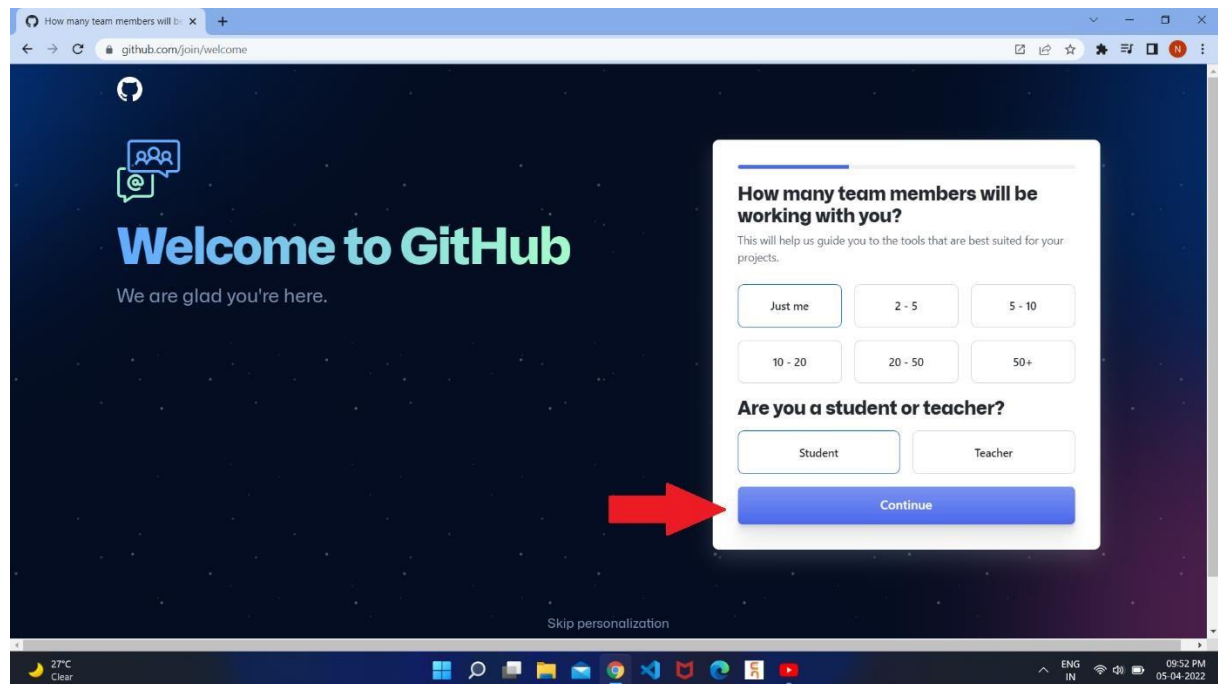
4. Configuring two-factor authentication

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.



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.



Collaborating on GitHub

Any number of people can work together in repositories across GitHub. You can configure settings, create project boards, and manage your notifications to encourage effective collaboration.

1. Working with repositories

Creating a repository

A repository is like a folder for your project. You can have any number of public and private repositories in your user account. Repositories can contain folders and files, images, videos, spreadsheets, and data sets, as well as the revision history for all files in the repository.

When you create a new repository, you should initialize the repository with a README file to let people know about your project.

2. Importing your projects

If you have existing projects, you'd like to move over to GitHub you can import projects using the GitHub Importer, the command line, or external migration tools.

3. Managing collaborators and permissions

You can collaborate on your project with others using your repository's issues, pull requests, and project boards. You can invite other people to your repository as collaborators from the **Collaborators** tab in the repository settings.

You are the owner of any repository you create in your user account and have full control of the repository. Collaborators have written access to your repository, limiting what they have permission to do.

4. Managing repository settings

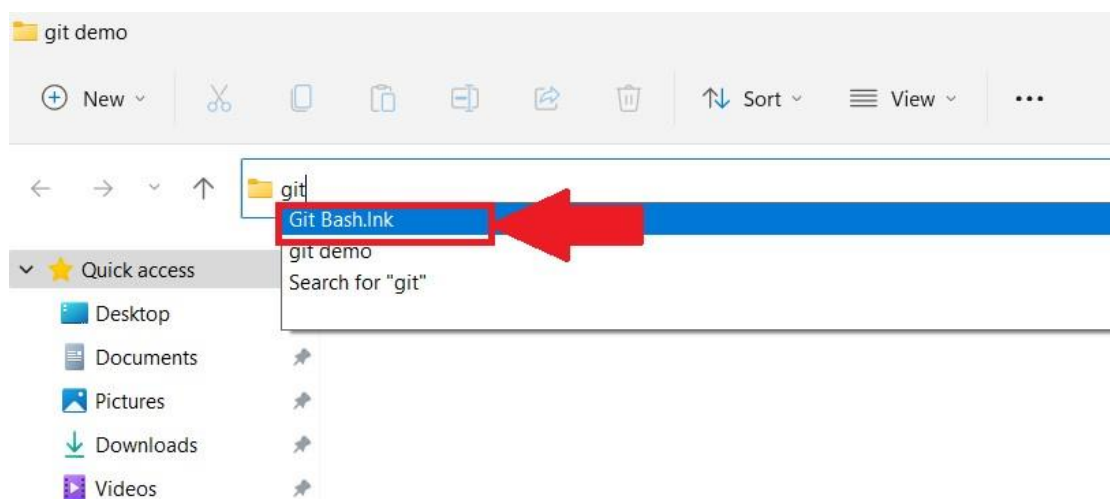
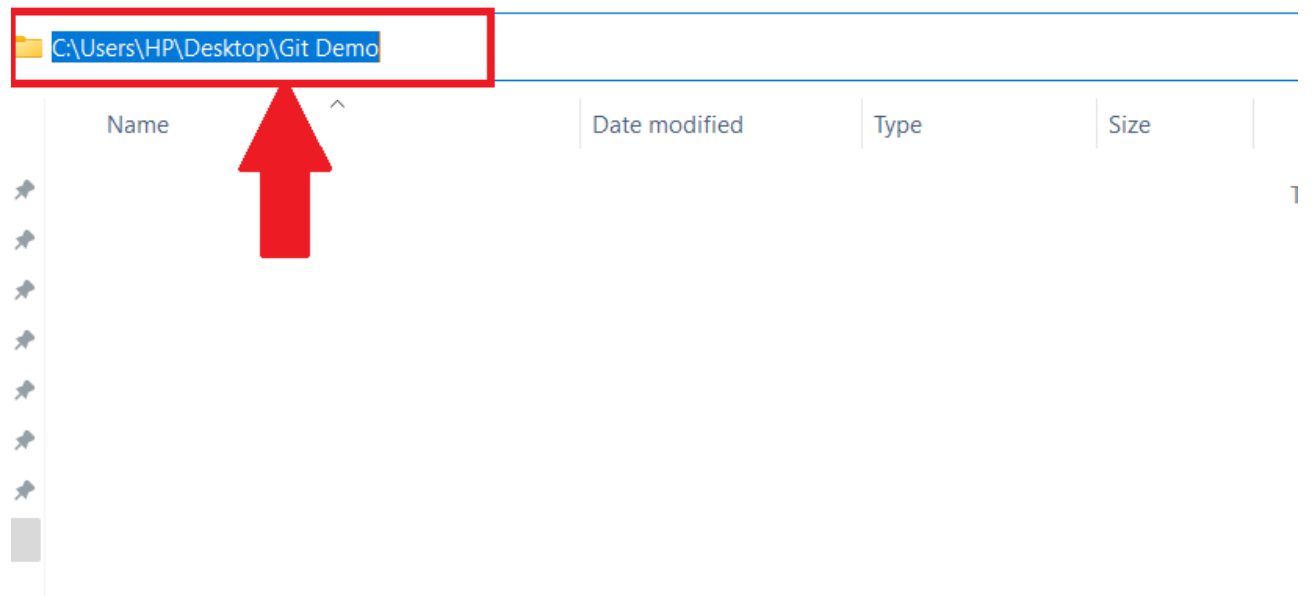
As the owner of a repository you can configure several settings, including the repository's visibility, topics, and social media preview.

3. Generate Logs

- Create a New Folder (Empty Folder).



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



- It will open Git Bash Terminal Or Git Bash CLI(Command Line Interface).
- Now we have to write all the Commands in this CLI.
- 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 initialized a new empty repository.*

```
HP@Abhishek MINGW64 ~/Desktop/Git Demo (master)
$ git init
Initialized empty Git repository in C:/Users/HP/Desktop/Git Demo/.git/
```

○ Type “pwd” , “git status” , “mkdir project-1”.

✚ **pwd (Present Work Directory):** *The pwd command writes the full pathname of the current working directory to the standard output.*

```
HP@Abhishek MINGW64 ~/Desktop/Git Demo (master)
$ pwd
/c/Users/HP/Desktop/Git Demo
```

✚ **git status :** *The git status command displays the state of the working directory and the staging area.*

```
HP@Abhishek MINGW64 ~/Desktop/Git Demo (master)
$ git status
On branch master

No commits yet

nothing to commit (create/copy files and use "git add" to track)
```

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

```
HP@Abhishek MINGW64 ~/Desktop/Git Demo (master)
$ mkdir project-1
```

- Type “cd project-1”, “touch file1”, “touch file2” and “ls”.

✚ **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.

```
HP@Abhishek MINGW64 ~/Desktop/Git Demo (master)
$ cd project-1
```

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

```
HP@Abhishek MINGW64 ~/Desktop/Git Demo/project-1 (master)
$ touch file1

HP@Abhishek MINGW64 ~/Desktop/Git Demo/project-1 (master)
$ touch file2
```

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

```
HP@Abhishek MINGW64 ~/Desktop/Git Demo/project-1 (master)
$ ls
file1 file2
```

- 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.

```
HP@Abhishek MINGW64 ~/Desktop/Git Demo/project-1 (master)
$ git add file1

HP@Abhishek MINGW64 ~/Desktop/Git Demo/project-1 (master)
$ git add file2
```

✚ **git add -A** : This command adds every change we made to files and folders from our repository to the Git staging area.

○ Now check git status, files are staged.

```
HP@Abhishek MINGW64 ~/Desktop/Git Demo/project-1 (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   file1
        new file:   file2
```

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

○ Open the Files in any editor like: “Notepad” and “VS code” and do changes in it and save it. You can also open file in notepad by command : **notepad <filename>**

```
HP@Abhishek MINGW64 ~/Desktop/Git Demo/project-1 (master)
$ notepad file1
```




```
HP@Abhishek MINGW64 ~/Desktop/Git Demo/project-1 (master)
$ notepad file2
```



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

```
HP@Abhishek MINGW64 ~/Desktop/Git Demo/project-1 (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   file1
        new file:   file2

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)
        modified:   file1
        modified:   file2
```

- The Files are modified, we need to add these modified files in staging area again to do Commits.
- 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 "UserName"***.

For userEmail : ***git config --global user.email "UserEmail"***.

```
HP@Abhishek MINGW64 ~/Desktop/Git Demo/project-1 (master)
$ git config --global user.name "Abhishek0060"

HP@Abhishek MINGW64 ~/Desktop/Git Demo/project-1 (master)
$ git config --global user.email "abhishek0060.be21@chitkara.edu.in"
```

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

```
HP@Abhishek MINGW64 ~/Desktop/Git Demo/project-1 (master)
$ git add -A
```

- 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.

```
HP@Abhishek MINGW64 ~/Desktop/Git Demo/project-1 (master)
$ git commit -m "This is my first commit"
[master (root-commit) 4a72b6f] This is my first commit
2 files changed, 2 insertions(+)
create mode 100644 project-1/file1
create mode 100644 project-1/file2
```

- 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.*

```
HP@Abhishek MINGW64 ~/Desktop/Git Demo/project-1 (master)
$ git log
commit 4a72b6fe045abfaf45fbf9e1c99613743563b1f8 (HEAD -> master)
Author: Abhishek0060 <abhishek0060.be21@chitkara.edu.in>
Date: Sat Apr 9 23:50:03 2022 +0530

    This is my first commit
```

4. 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, which gives us List of all branches in our Repository. If the branch name is in **Green Colour**, it show the branch in which you are.

```
HP@Abhishek MINGW64 ~/Desktop/Git Demo/project-1 (master)
$ git branch
* master
```

○ Write command “**git branch trial**” on CLI.

✚ **git branch <branch name>** : This Command Allow us to make a new Branch (which is Copy of Master branch).

```
HP@Abhishek MINGW64 ~/Desktop/Git Demo/project-1 (master)
$ git branch
* master

HP@Abhishek MINGW64 ~/Desktop/Git Demo/project-1 (master)
$ git branch trial

HP@Abhishek MINGW64 ~/Desktop/Git Demo/project-1 (master)
$ git branch
* master
  trial
```

○ Type “**git checkout trial**”.

✚ **git Checkout <branch name>** : This command allow us to Switch to another Branch (i.e.) trial.

```
HP@Abhishek MINGW64 ~/Desktop/Git Demo/project-1 (master)
$ git checkout trial
Switched to branch 'trial'
```

The trial branch is the copy of Master branch. It contains all the Commits done in master branch.

If we do changes in the trial branch it will not affect our master branch.

- Creating a new file “trial_file” in trial branch and add it in staged area.

```
HP@Abhishek MINGW64 ~/Desktop/Git Demo/project-1 (master)
$ git checkout trial
Switched to branch 'trial'

HP@Abhishek MINGW64 ~/Desktop/Git Demo/project-1 (trial)
$ touch trial_file

HP@Abhishek MINGW64 ~/Desktop/Git Demo/project-1 (trial)
$ git add trial_file

HP@Abhishek MINGW64 ~/Desktop/Git Demo/project-1 (trial)
$ git status
On branch trial
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   trial_file
```

- Do changes in the “trial_file” and commit it.

```
HP@Abhishek MINGW64 ~/Desktop/Git Demo/project-1 (trial)
$ notepad trial_file
```



```
HP@Abhishek MINGW64 ~/Desktop/Git Demo/project-1 (trial)
$ git add trial_file

HP@Abhishek MINGW64 ~/Desktop/Git Demo/project-1 (trial)
$ git commit -m "This commit is for trial_file"
[trial bf75da9] This commit is for trial_file
1 file changed, 1 insertion(+)
create mode 100644 project-1/trial_file

HP@Abhishek MINGW64 ~/Desktop/Git Demo/project-1 (trial)
$ git status
On branch trial
nothing to commit, working tree clean
```

Now Switch to Master branch.

```
HP@Abhishek MINGW64 ~/Desktop/Git Demo/project-1 (trial)
$ git checkout master
Switched to branch 'master'

HP@Abhishek MINGW64 ~/Desktop/Git Demo/project-1 (master)
$ git log
commit 4a72b6fe045abfaf45fbf9e1c99613743563b1f8 (HEAD -> master)
Author: Abhishek0060 <abhishek0060.be21@chitkara.edu.in>
Date: Sat Apr 9 23:50:03 2022 +0530

    This is my first commit
```

We can see that after switching to master branch and check git log, it doesn't show "trial" and only shows the commit done in master 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,

- On CLI, first switch to master branch.

- Type Command "**git merge trial**".

- ✚ **git merge <branch Name>** : It merge the branch into another branch.

The changes or added data merge in master branch.

5. 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:

1. Working directory
2. Staging area
3. 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 have 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. You can find your staging area in the `.git` folder inside the index file.

Currently, the file2 which is a text file here is untracked and so it is red in colour.

In other words, staging area is the playground where you group, add and organize the files to be committed to Git for tracking their versions.

Indexing is the process of adding files to the staging area. In other words, index constitutes of files added to the staging area. It is the intermediate stage from where files are passed to be committed and are converted from red colour to green colour.

Note: `git add` - Command to add files to staging area.

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.

Thus, Git directory is the database where metadata about project files' history will be tracked.

Note: `git commit -m "your message"` - Command to commit files to Git repository with message.

