

Git-Windows

Download Git for windows

1. Navigate to the website <https://git-scm.com/downloads>
2. Click the download link for Windows and allow the download to complete.

The image shows two screenshots of the Git website. The top screenshot is the main 'Downloads' page, and the bottom screenshot is the 'Download for Windows' page.

Top Screenshot: Git Downloads Page

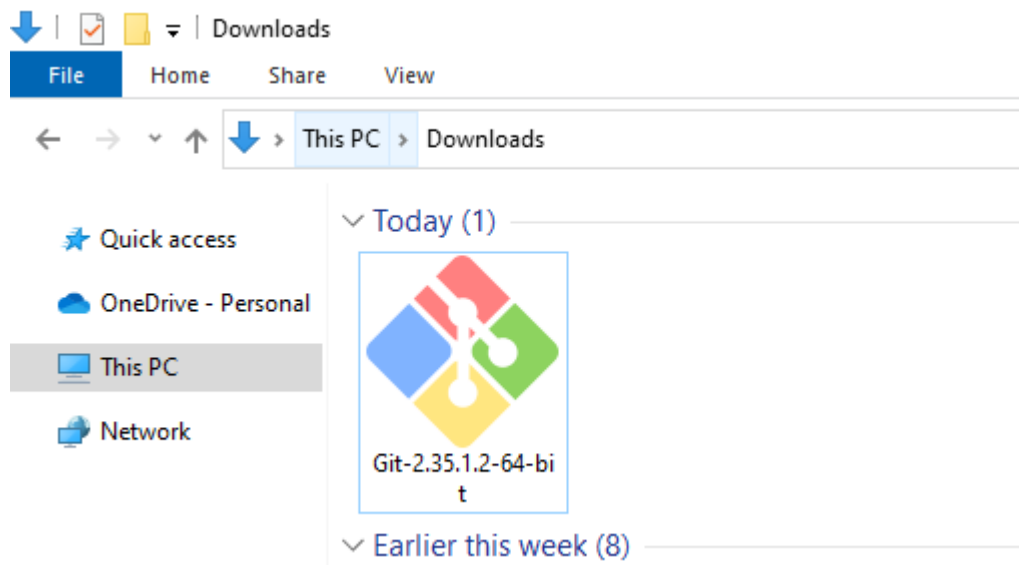
- The browser address bar shows git-scm.com/downloads.
- The page header includes the Git logo and the tagline "--distributed-is-the-new-centralized".
- A search bar is located in the top right corner.
- The left sidebar contains links: About, Documentation, Downloads (highlighted in red), GUI Clients, Logos, and Community.
- The main content area is titled "Downloads". It features three platform buttons: macOS, Windows (highlighted with a red box and a red arrow pointing to it), and Linux/Unix.
- To the right of the platform buttons, there is a section for the "Latest source Release" (2.35.1) with a "Download for Windows" button.
- Below the platform buttons, there is a note: "Older releases are available and the Git source repository is on GitHub."
- At the bottom, there are sections for "GUI Clients" and "Logos".

Bottom Screenshot: Download for Windows Page

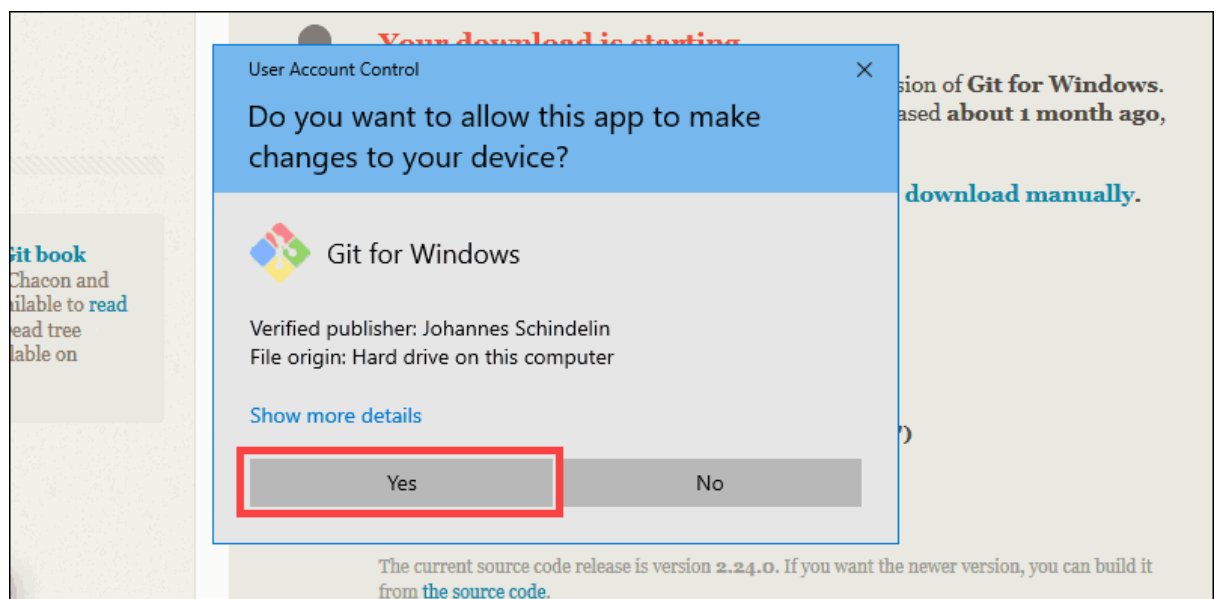
- The browser address bar shows git-scm.com/downloads.
- The page header includes the Git logo and the tagline "--fast-version-control".
- A search bar is located in the top right corner.
- The left sidebar contains links: About, Documentation, Downloads (highlighted in red), GUI Clients, Logos, and Community.
- The main content area is titled "Download for Windows".
- Below the title, there is a link "Click here to download" (highlighted with a red box and a red arrow pointing to it) followed by the text: "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."
- Below this, there is a section titled "Other Git for Windows downloads" with links for "Standalone Installer", "32-bit Git for Windows Setup", "64-bit Git for Windows Setup", "Portable ('thumbdrive edition')", "32-bit Git for Windows Portable", and "64-bit Git for Windows Portable".

Extract and Launch Git Installer

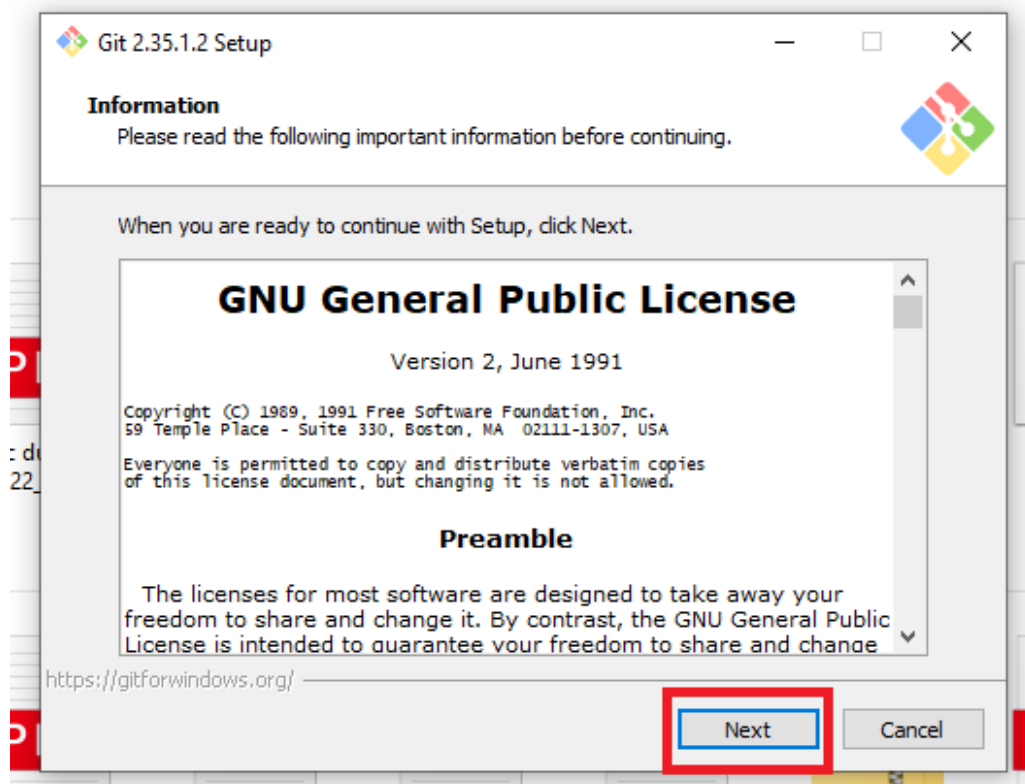
3. Browse to the download location (or use the download shortcut in your browser). Double-click the file to extract and launch the installer.



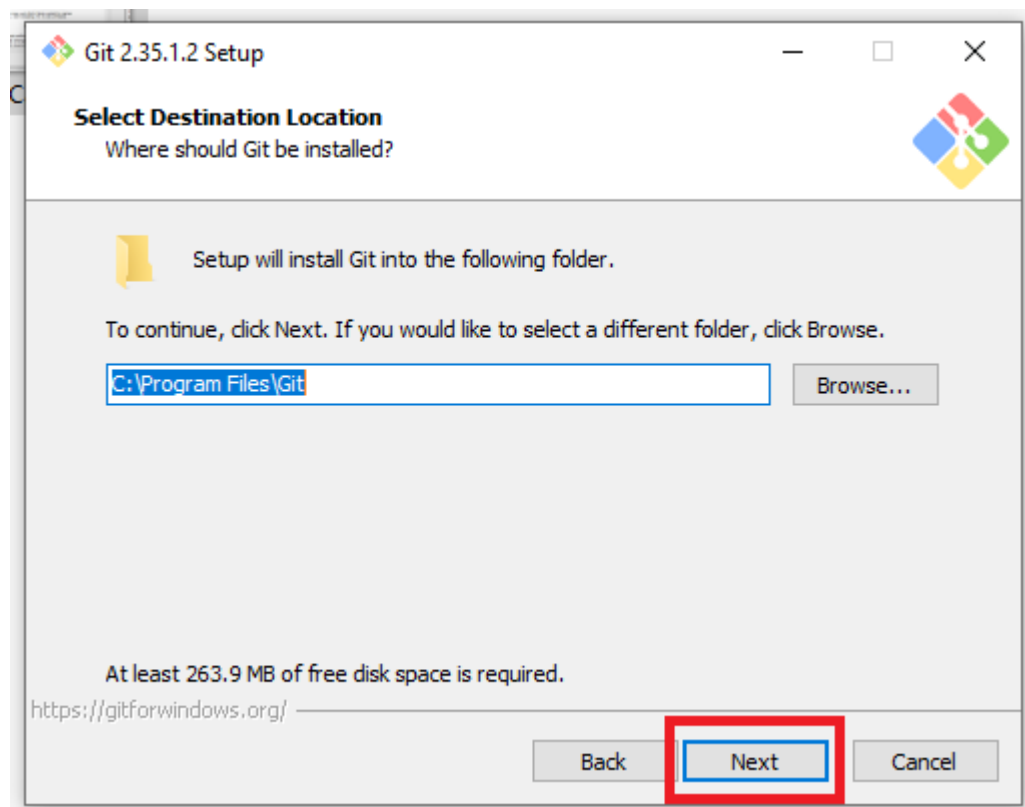
4. Allow the app to make changes to your device by clicking Yes on the User Account Control dialog that opens.



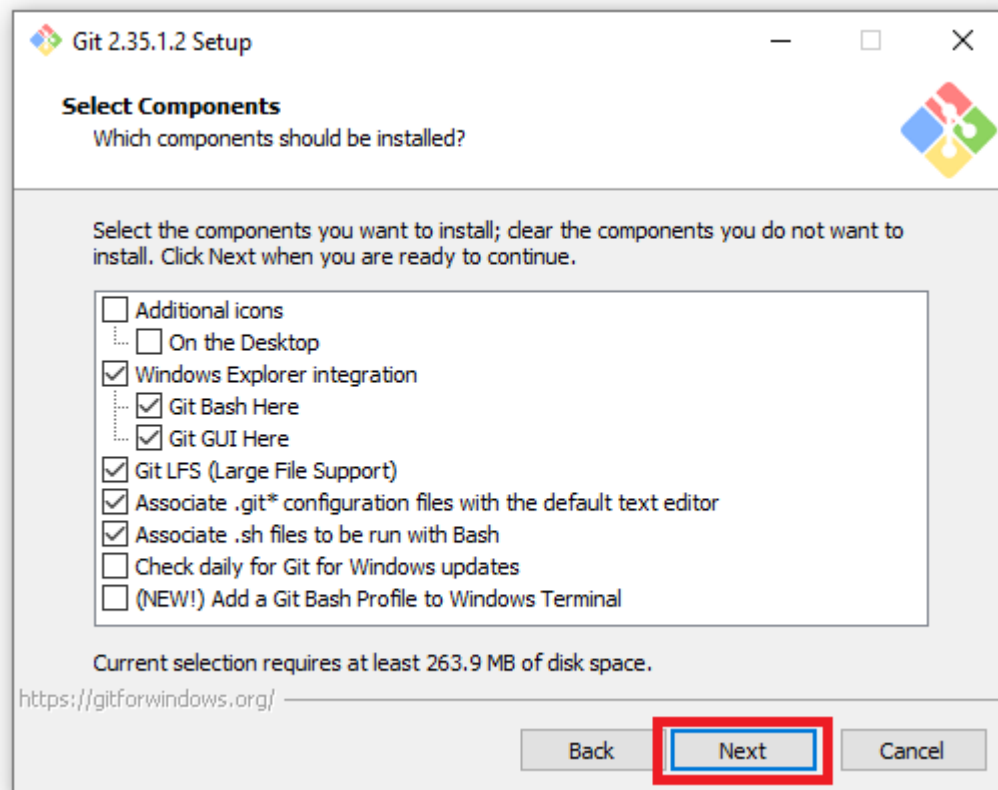
5. Review the GNU General Public License, and when you're ready to install, click Next.



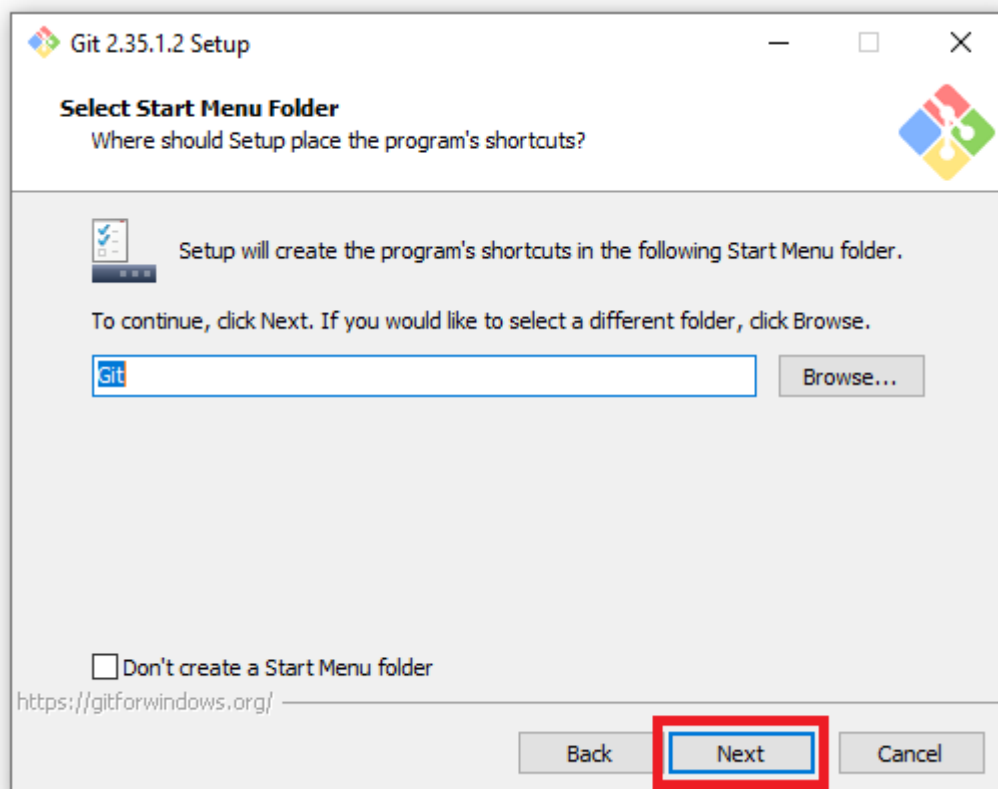
6. The installer will ask you for an installation location. Leave the default, unless you have reason to change it, and click Next.



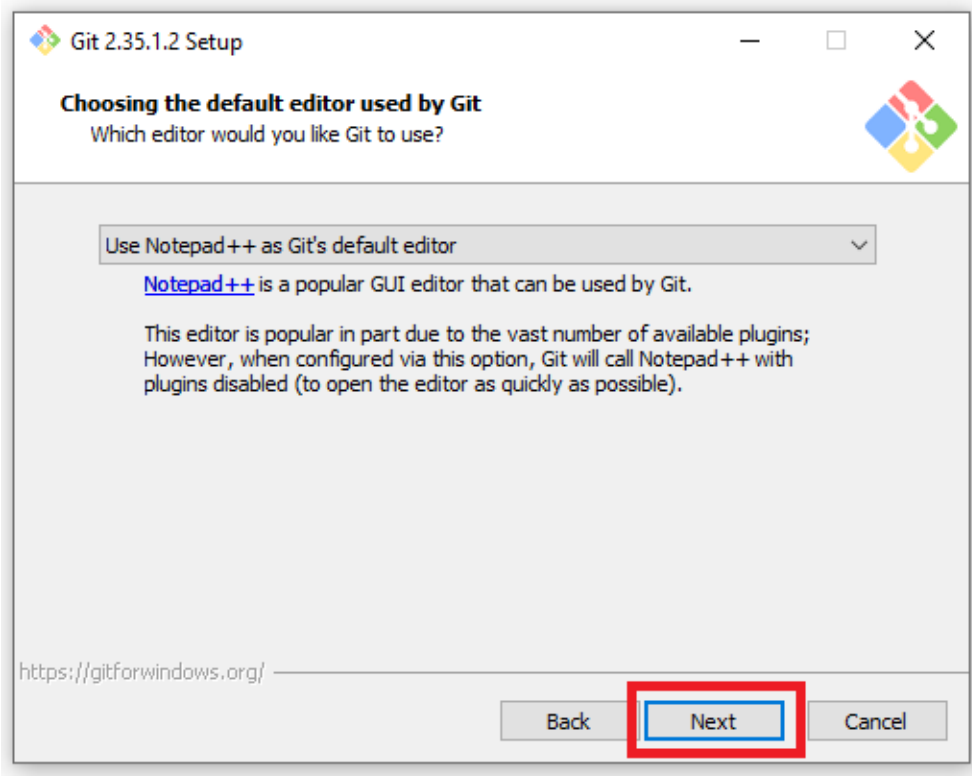
7. A component selection screen will appear. Leave the defaults unless you have a specific need to change them and click Next.



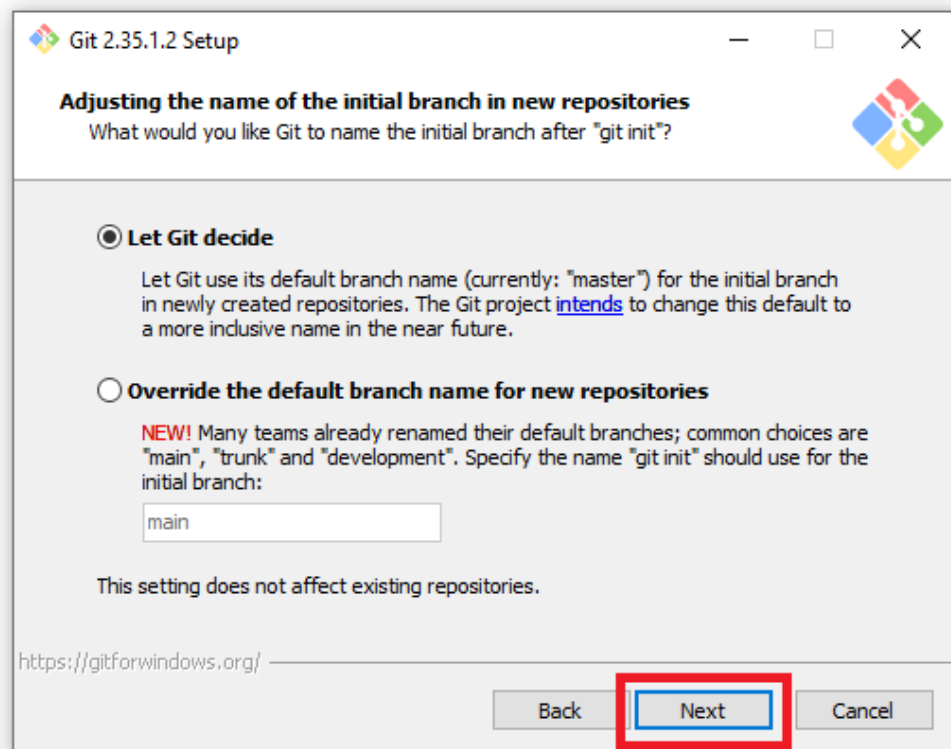
8. The installer will offer to create a start menu folder. Simply click Next.



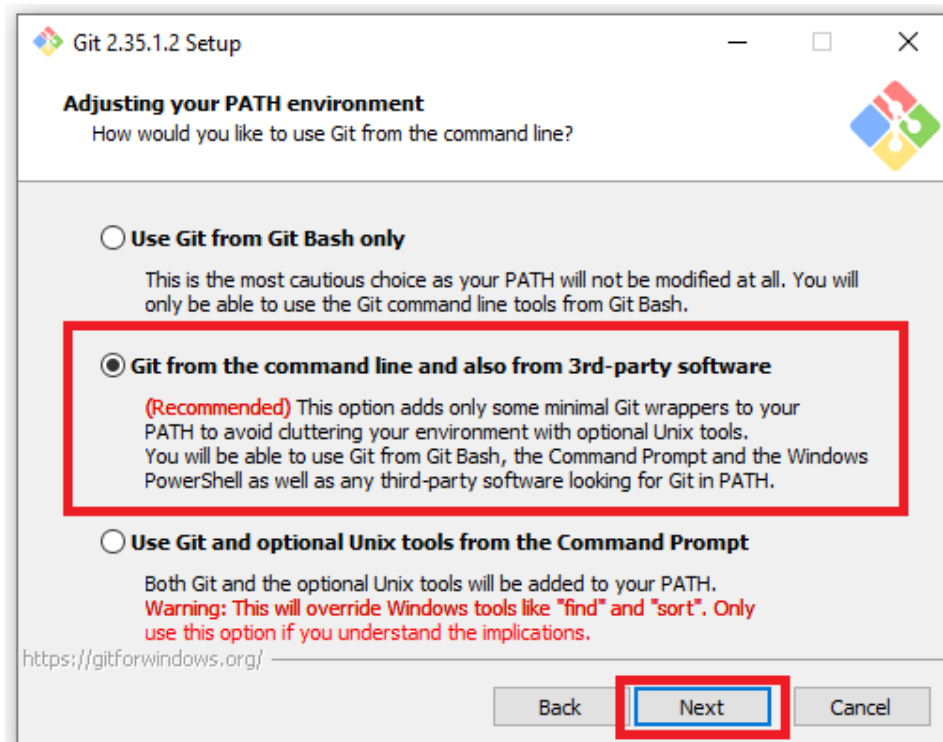
9. Select a text editor you'd like to use with Git. Use the drop-down menu to select Notepad++ (or whichever text editor you prefer) and click Next.



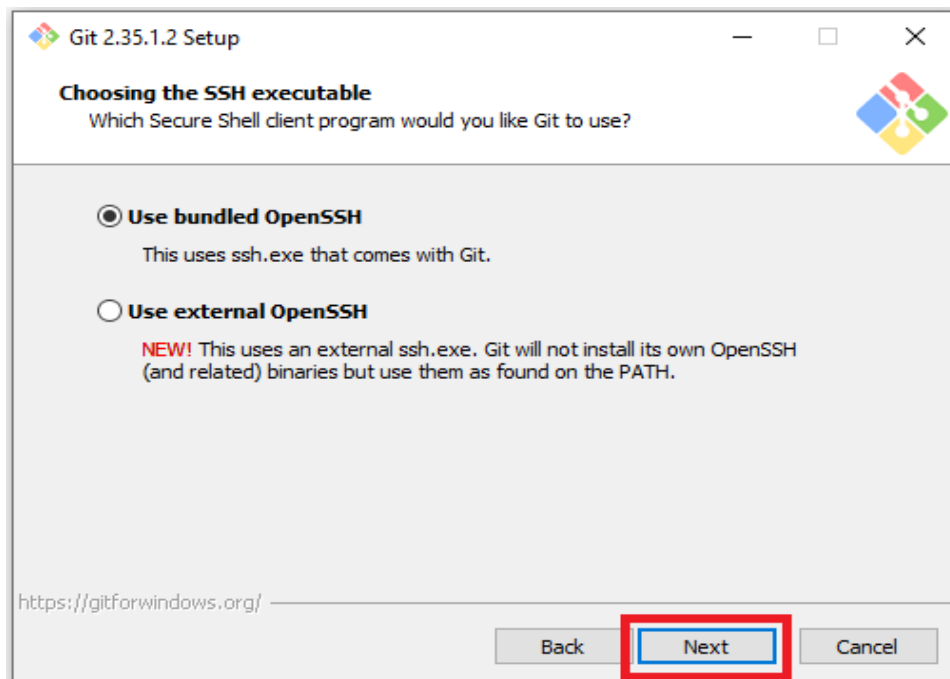
10. The next step allows you to choose a different name for your initial branch. The default is 'master.' Unless you're working in a team that requires a different name, leave the default option and click Next.



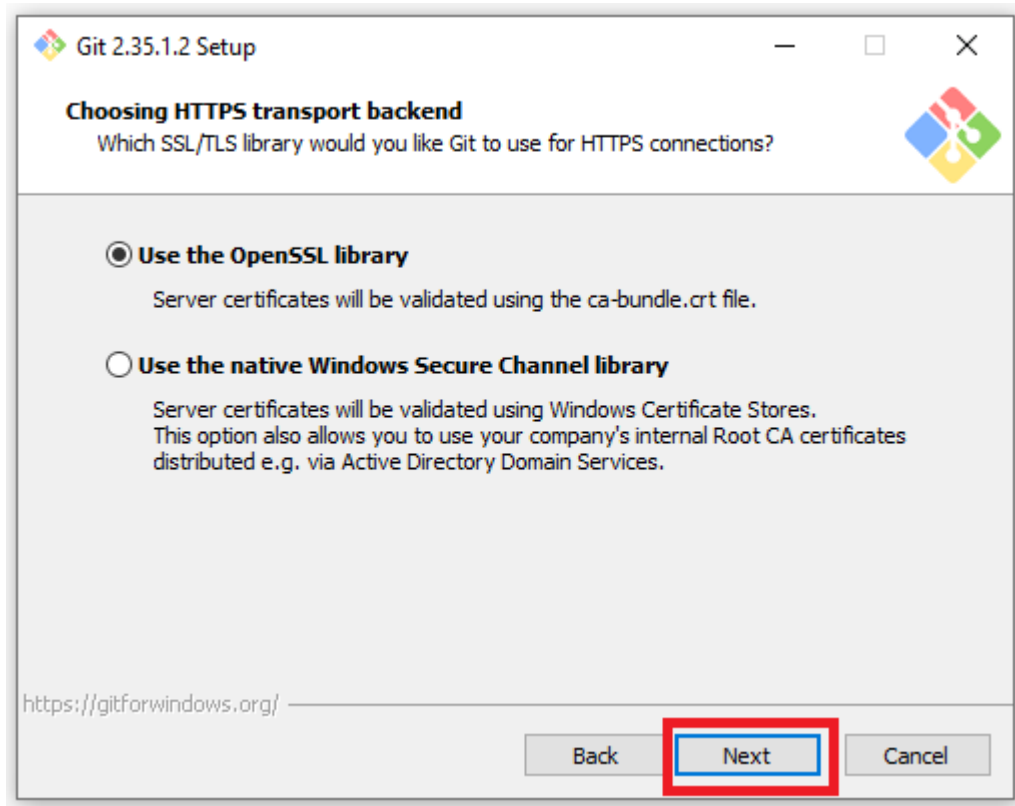
11. This installation step allows you to change the PATH environment. The PATH is the default set of directories included when you run a command from the command line. Leave this on the middle (recommended) selection and click Next



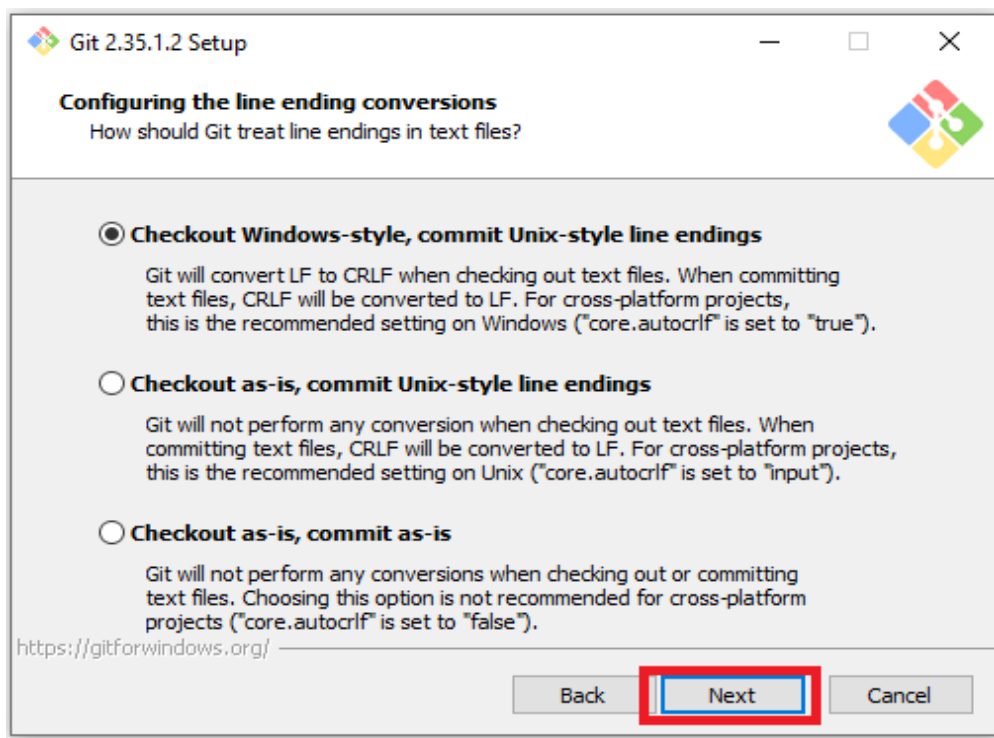
12. The installer now asks which SSH client you want Git to use. Git already comes with its own SSH client, so if you don't need a specific one, leave the default option and click Next.



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.



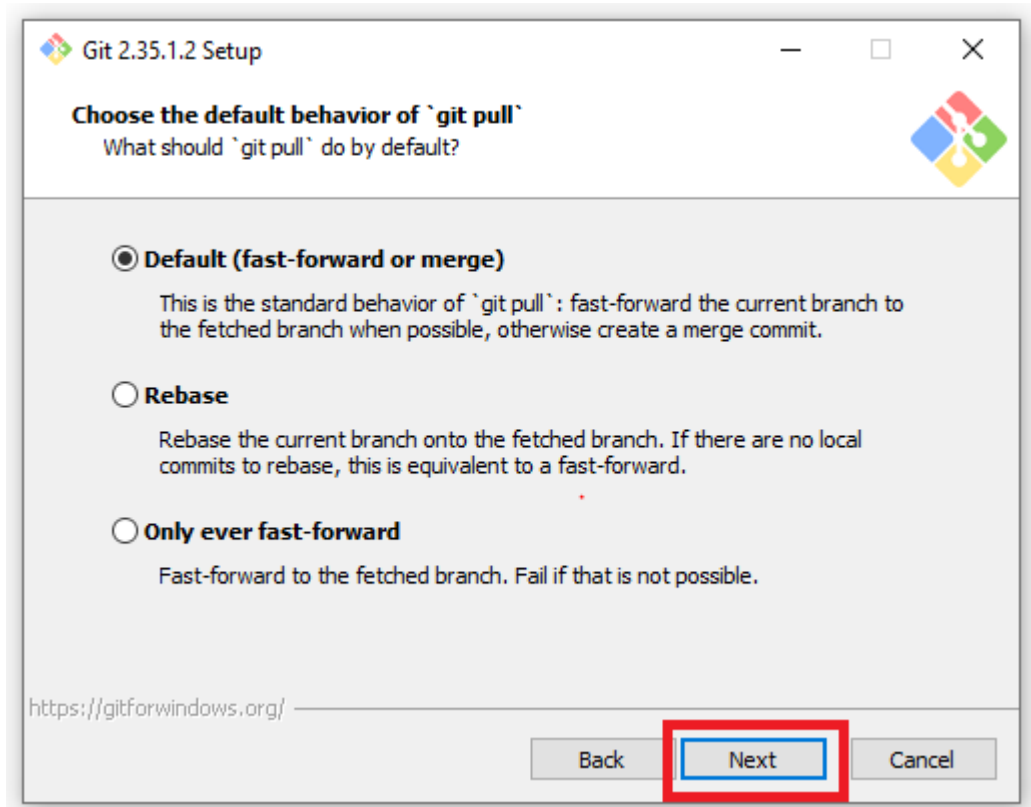
14. The next selection converts line endings. It is recommended that you leave the default selection. This relates to the way data is formatted and changing this option may cause problems. Click Next.



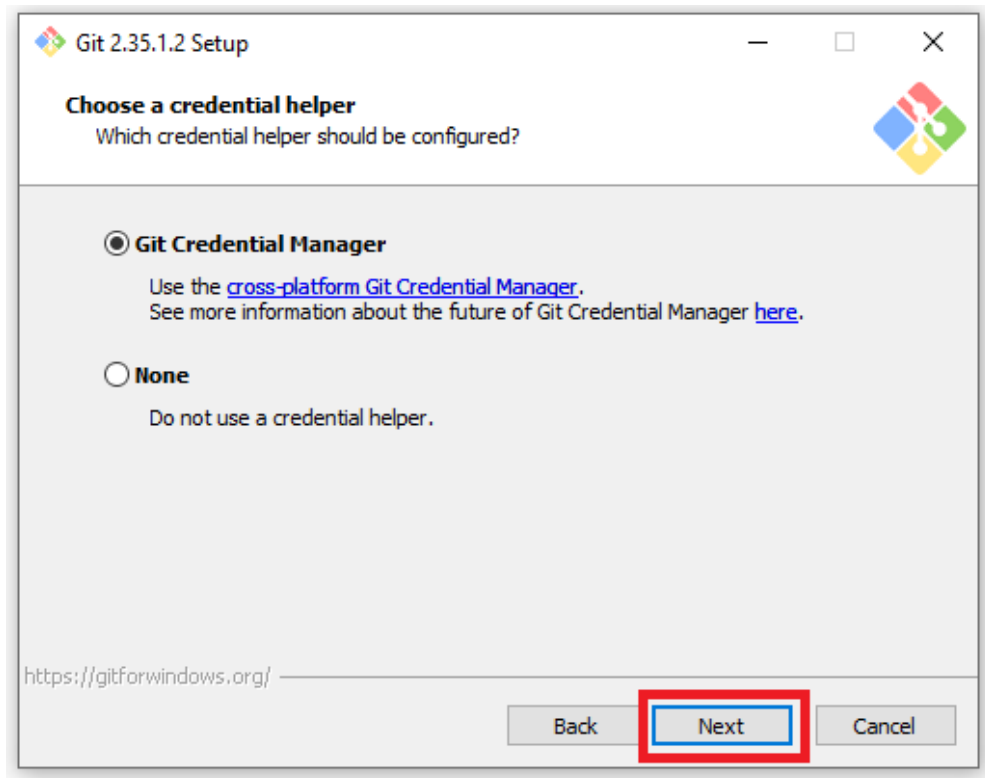
15. Choose the terminal emulator you want to use. The default MinTTY is recommended, for its features. Click Next.



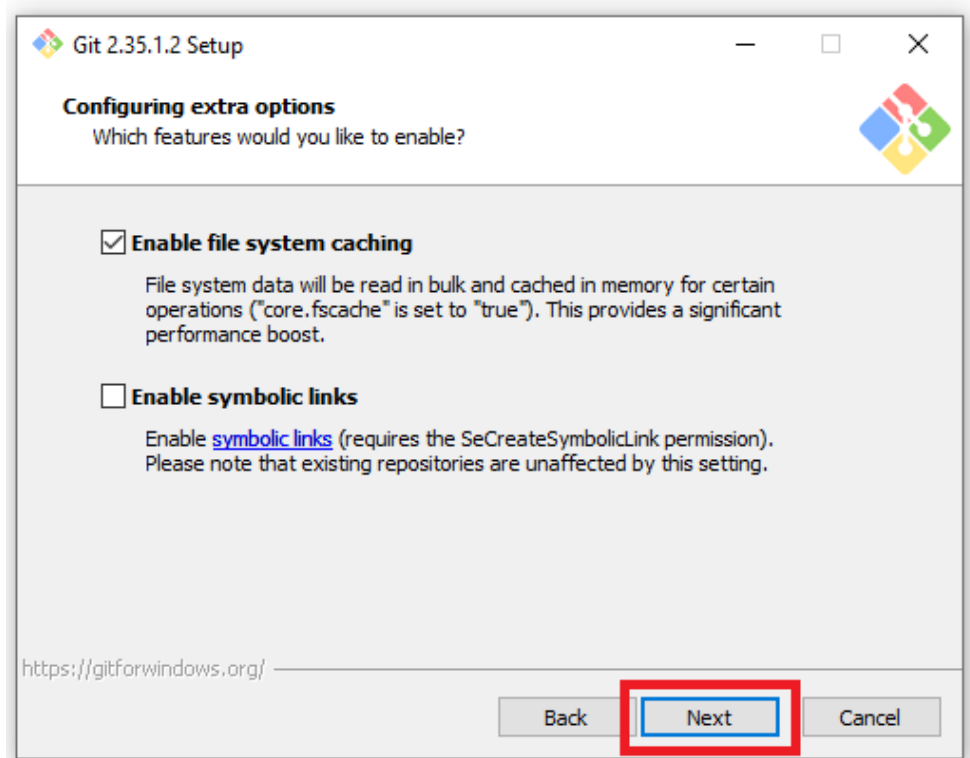
16. The installer now asks what the git pull command should do. The default option is recommended unless you specifically need to change its behaviour. Click Next to continue with the installation.



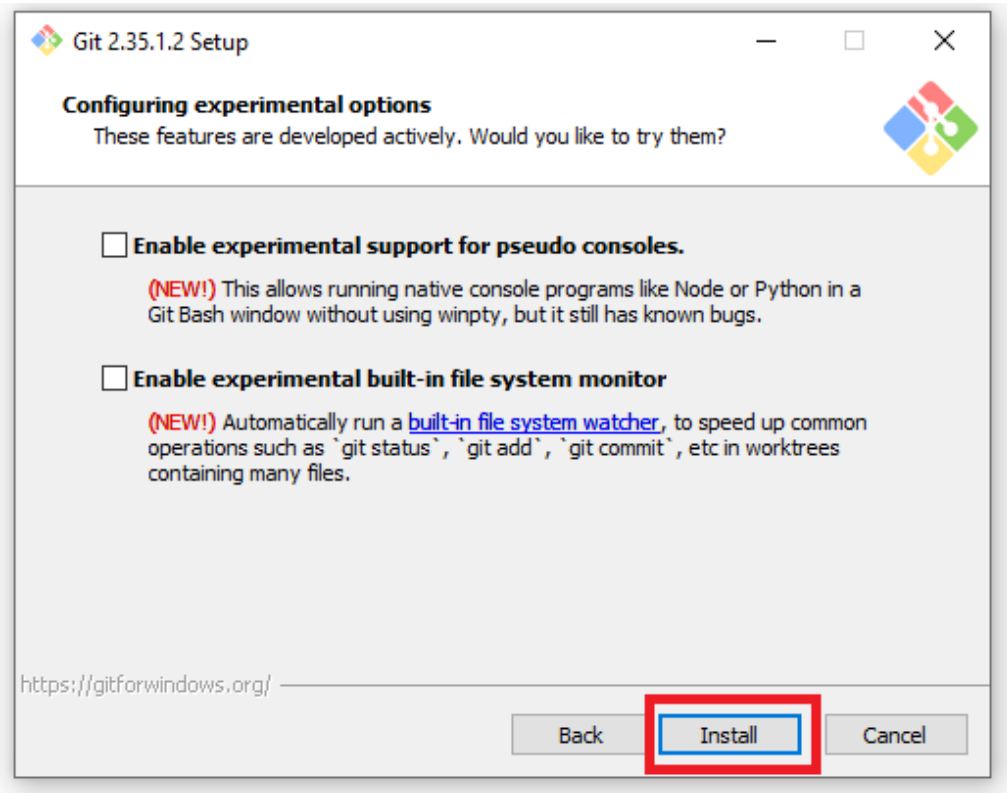
17. Next you should choose which credential helper to use. Git uses credential helpers to fetch or save credentials. Leave the default option as it is the most stable one, and click Next.



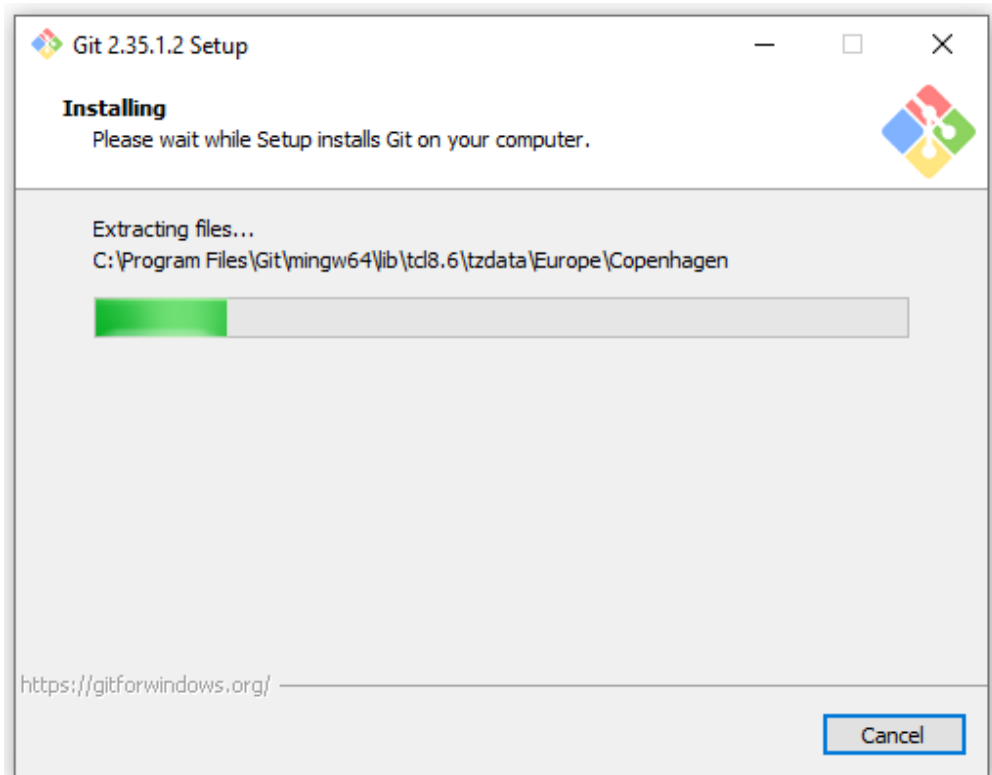
18. The default options are recommended, however this step allows you to decide which extra option you would like to enable. If you use symbolic links, which are like shortcuts for the command line, tick the box. Click Next



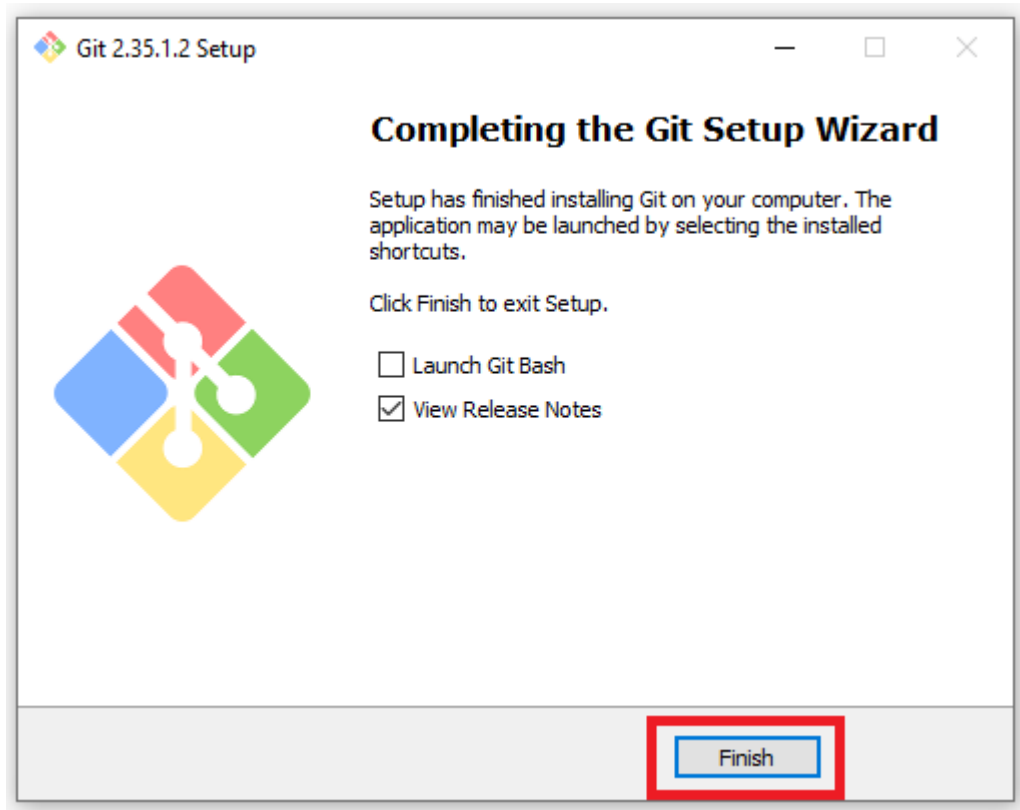
19. Depending on the version of Git you're installing, it may offer to install experimental features. At the time this manual was written, the options to include support for pseudo controls and a built-in file system monitor were offered. Unless you are feeling adventurous, leave them unchecked and click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



20. Wait for the installation to complete.



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



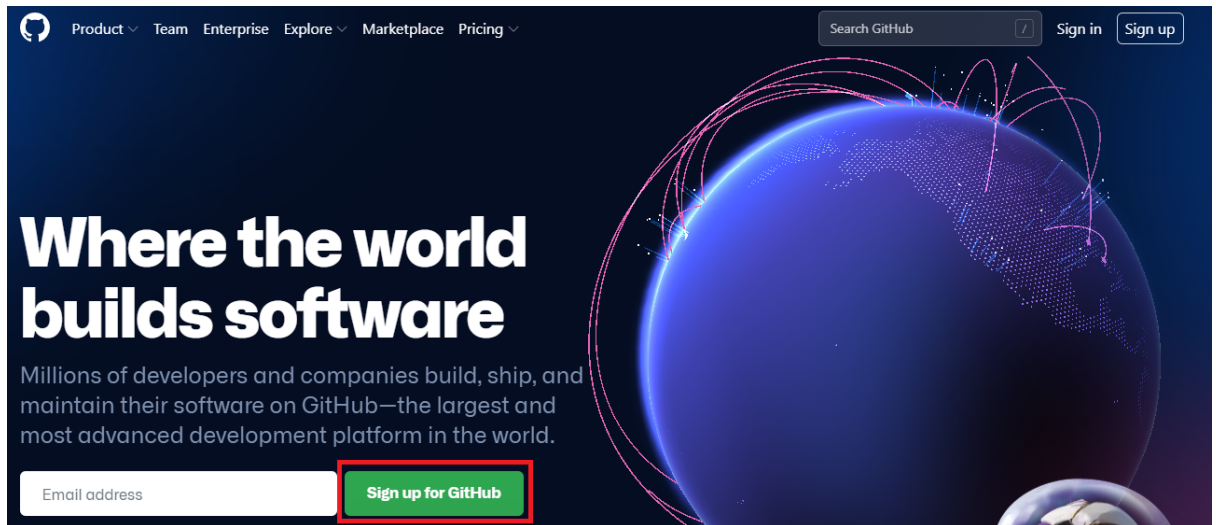
22. To check if git is installed properly, go to the command prompt and type “*git --version*” then you should see the git version.

The image shows a Windows Command Prompt window. The title bar reads 'Command Prompt'. The text inside the window shows the standard Windows version information: 'Microsoft Windows [Version 10.0.19044.1586] (c) Microsoft Corporation. All rights reserved.' followed by the command prompt 'C:\Users\...>'. The user has entered the command 'git --version', and the output displayed is 'git version 2.35.1.windows.2'.

23. Now you have to create a github account. GitHub is a software development platform that allows you to save, track, and collaborate on software projects online. It allows developers to upload their own code files and work on open-source projects with other developers.

24. Navigate to the website <https://github.com>

25. If you already have an account you can directly **Sign in**, Otherwise to create a new account choose **Sign Up for Github**.

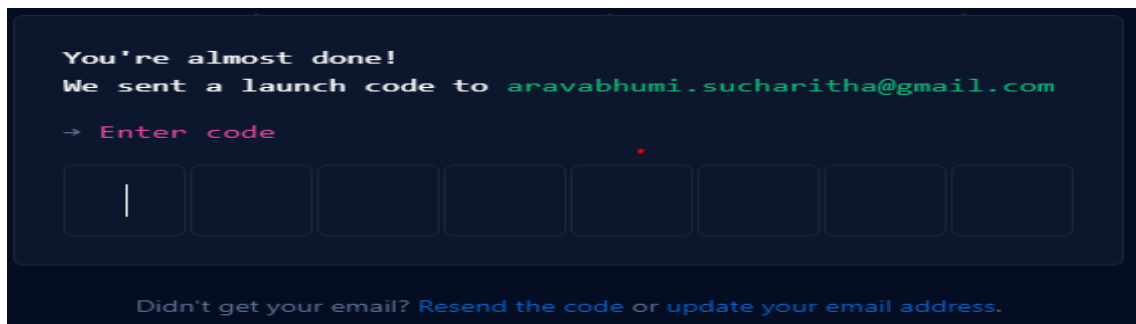


Fill all the details

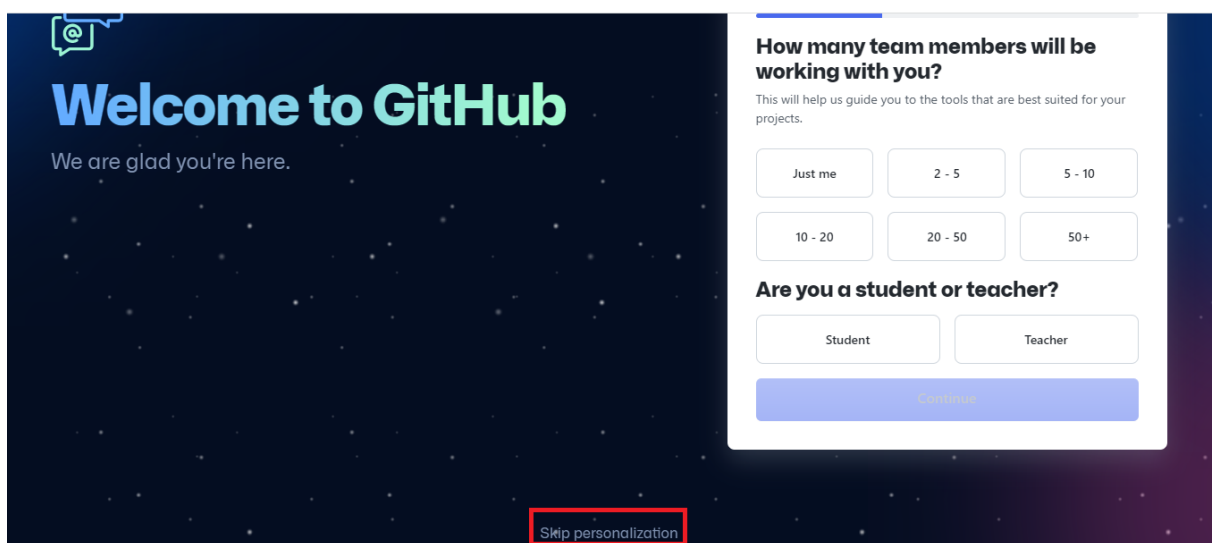
The image shows the GitHub sign-up form. It has a dark blue background with white text. The form contains the following elements:

- A welcome message: 'Welcome to GitHub! Let's begin the adventure'.
- A section titled 'Enter your email' with a red 'X' icon indicating a validation error.
- A section titled 'Create a password' with a red 'X' icon indicating a validation error.
- A section titled 'Enter a username' with a red arrow icon indicating a validation error.
- A 'Continue' button.
- A question: 'Would you like to receive product updates and announcements via email?'.
- A prompt: 'Type "y" for yes or "n" for no'.
- A green checkmark icon followed by the letter 'n', indicating the user has accepted the terms.

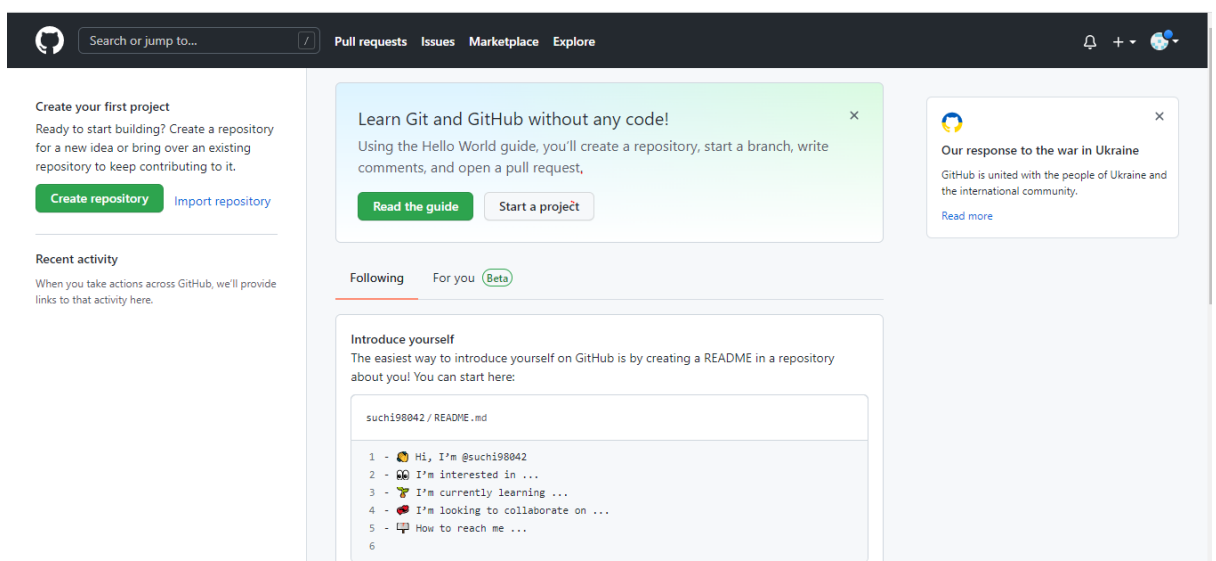
26. You will get a registration code to the email used for registering to github



27. You have to answer a few personalization questions or you can choose skip personalization .



28. You are all set to work on github, after you're done registering you will be redirected to the dashboard page of github.



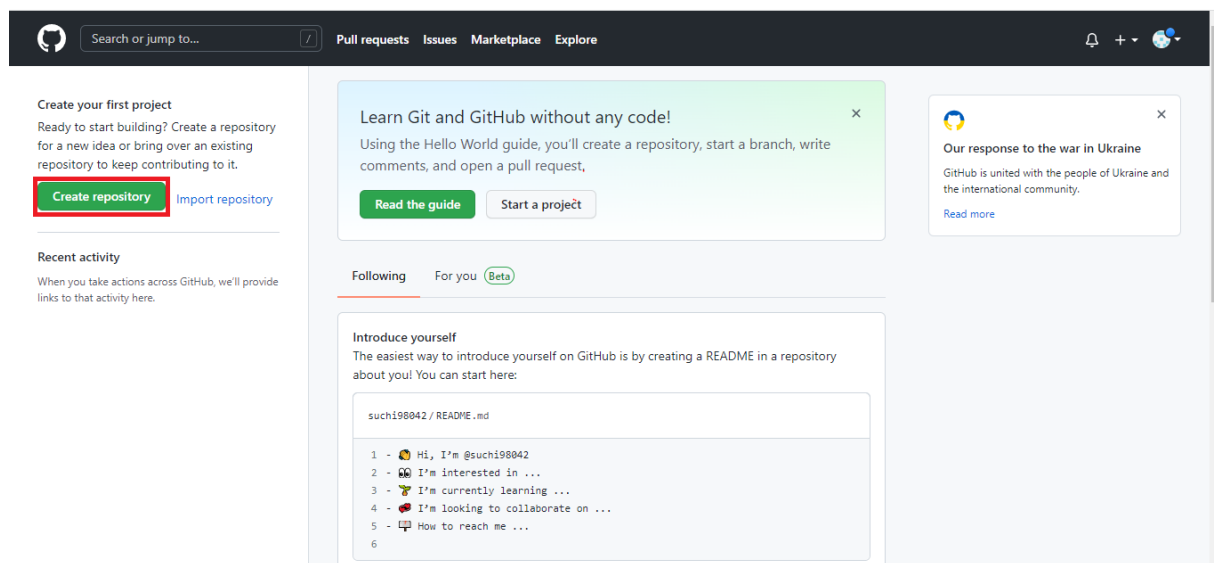
29. You have git for windows and a github account. You have to configure the git before using it. Open command prompt and type following commands.

```
git config --global user.name "your username here"
```

```
git config --global user.email "your email here"
```

To Create Repository-

1. Navigate to the website <https://github.com>
2. **Sign In** with your credentials and select the option **Create repository**.




3. Type the name of your repository and description is optional.

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Owner ^{*} Repository name ^{*}


 suchi98042 ▾ / Trial_repo ✓


Great repository names are short and memorable. Need inspiration? How about [scaling-palm-tree?](#)

Description (optional)

4. Next section you choose if your repository is public or private. If you choose public, anyone on the internet can see your code but code commits you can

decide. If you choose private, only people you choose can see the code and make any commits

☒  **Public**
Anyone on the internet can see this repository. You choose who can commit.

☐  **Private**
You choose who can see and commit to this repository.

5. Next step you will be shown various options

Add a README file- This option allows you to add a README file for your repository. README is a markdown file where you write a long description for your project and add any instructions or documentation that you want to share with others. Use Markdown to format headings, lists, links, etc., I suggest you add this file when creating the repository as you can keep editing this file as you add more functionalities to your project. Example of README.md file- left side shows how the text file looks, right side shows how it is displayed on github.

<pre>Fooobar is a Python library for dealing with word pluralization. ## Installation Use the package manager [pip](https://pip.pypa.io/en/stable/) to install foobar. ```bash pip install foobar ``` ## Usage ```python import foobar # returns 'words' foobar.pluralize('word') # returns 'geese' foobar.pluralize('goose') # returns 'phenomenon' foobar.singularize('phenomena') ``` ## Contributing Pull requests are welcome. For major changes, please open an issue first to discuss what you would like to change. Please make sure to update tests as appropriate.</pre>	<div>Fooobar is a Python library for dealing with word pluralization.</div> <div>Installation</div> <div>Use the package manager pip to install foobar.</div> <div><pre>pip install foobar</pre></div> <div>Usage</div> <div><pre>import foobar # returns 'words' foobar.pluralize('word') # returns 'geese' foobar.pluralize('goose') # returns 'phenomenon' foobar.singularize('phenomena')</pre></div> <div>Contributing</div> <div>Pull requests are welcome. For major changes, please open an issue first to discuss what you would like to change.</div>
---	---

Add .gitignore- This option allows you to add a .gitignore file for your repository. The .gitignore file is a text file that tells Git which files or folders to ignore in a project. I suggest you add this file when creating the repository. You can choose a template from github or you can write on your own. Example of .gitignore file

```
.gitignore - Notepad
File Edit Format View Help

# Visual Studio Code
.vscode

# User-specific files
*.suo
*.user
*.userosscache
*.sln.docstates

# Build results
[Dd]ebug/
[Dd]ebugPublic/
[Rr]elease/
[Rr]eleases/
x64/
x86/
bld/
[Bb]in/
[Oo]bj/
msbuild.log
msbuild.err
msbuild.wrn
```

Choose a license- A license tells others what they can and can't do with your code.

Initialize this repository with:

Skip this step if you're importing an existing repository.

☒ Add a README file

This is where you can write a long description for your project. [Learn more.](#)


☒ Add .gitignore

Choose which files not to track from a list of templates. [Learn more.](#)

.gitignore template: C++ ▼

☐ Choose a license

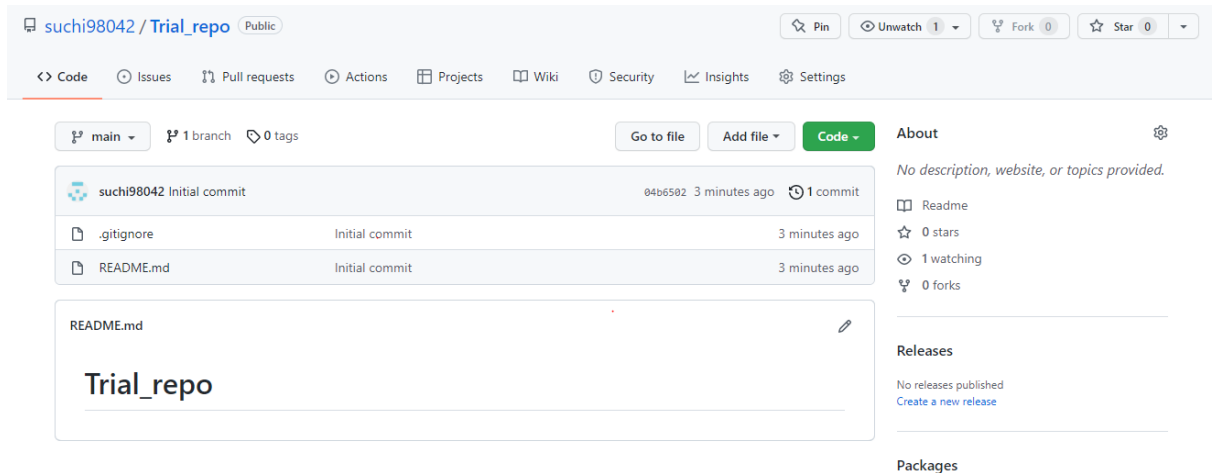
A license tells others what they can and can't do with your code. [Learn more.](#)

This will set  main as the default branch. Change the default name in your [settings](#).

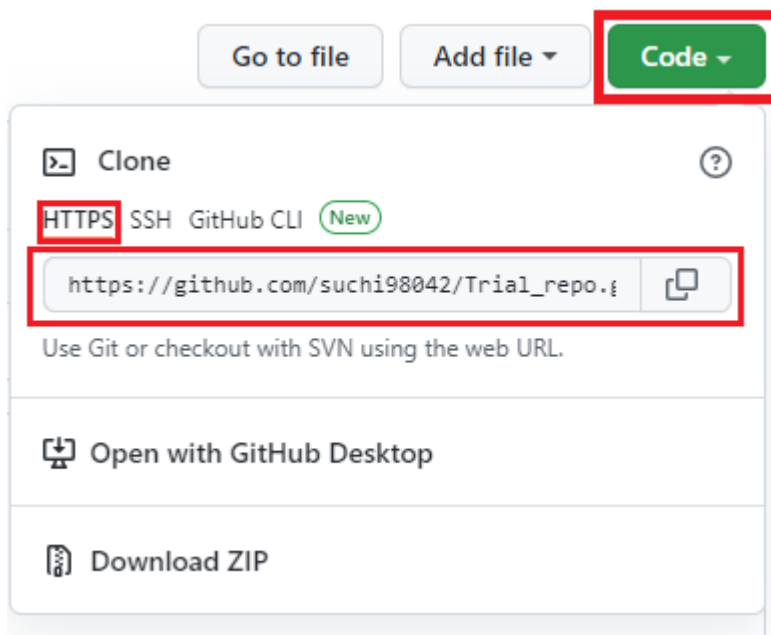
6. Then select **Create repository**. By default it will create a repository with one branch which is named either master or main branch.

Clone repository from github-

1. Navigate to the repository you would like to clone. A repository will look like this-



2. Select the option **code** on the right hand side corner, select **HTTPS** and copy the link. The default option is **HTTPS**.



3. Open command prompt and navigate to the location on your computer in which you want to clone the repository. Type command *git clone link we copied in the previous step.*




```
git clone https://github.com/suchi98042/Trial_repo.git
```

4. To clone a particular branch *git clone -b branch-name link from repo.'*

5. After cloning it creates a folder with your repo name. Navigate to the folder you can see all the files in your repository in the folder. You can work on the files as you work regularly.

```
Cloning into 'Trial_repo'...
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (4/4), done.
```

] Name ^

-  .git
-  .gitignore
-  README.md

6. *git branch* command shows the branch which you cloned into the local repository.

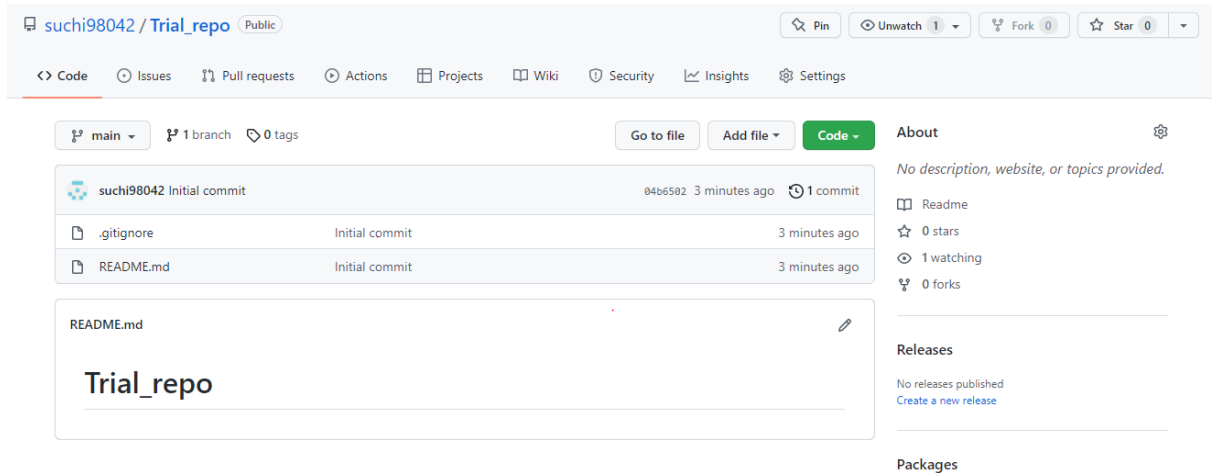
```
C:\Users\Sai Sucharitha Arava\OneDrive\Documents\trial>cd Trial_repo
C:\Users\Sai Sucharitha Arava\OneDrive\Documents\trial\Trial_repo>git branch
* main
```

Creating a new branch-

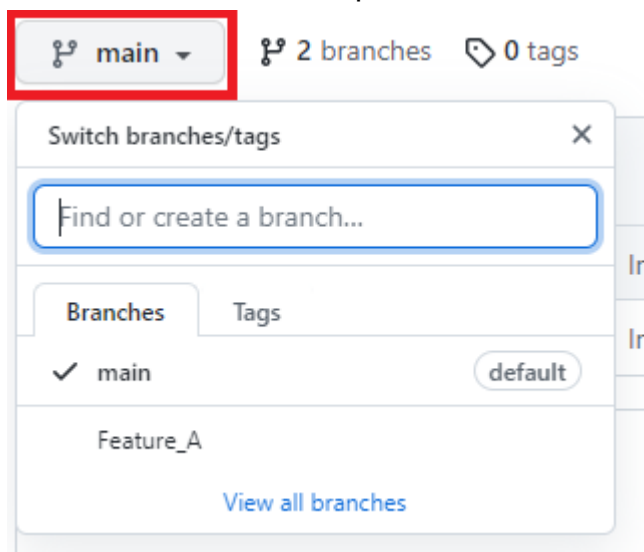
There are two ways to create a branch from github and the second way is from command prompt.

From github:

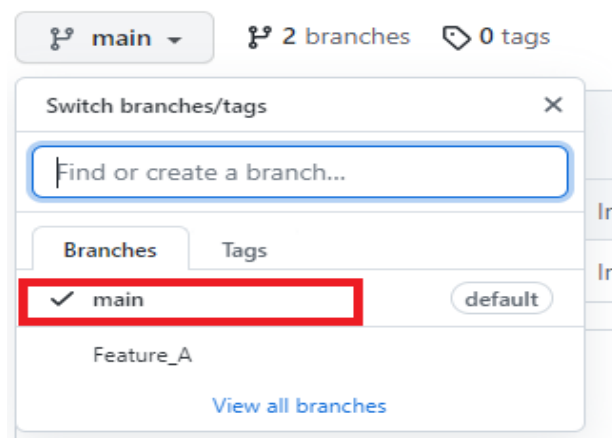
1. Navigate to the repository you would like to create a branch. A repository will look like this-



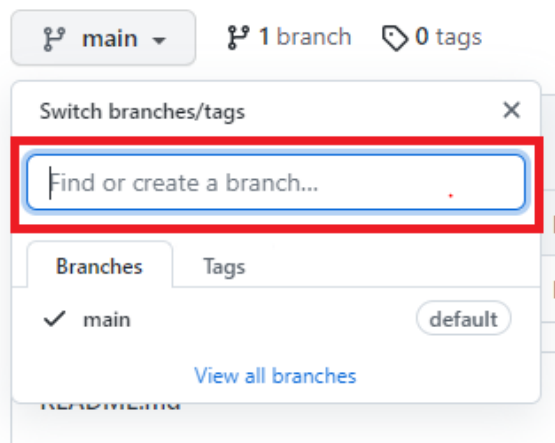
2. Select the branch list dropdown from the left hand side of your dashboard.



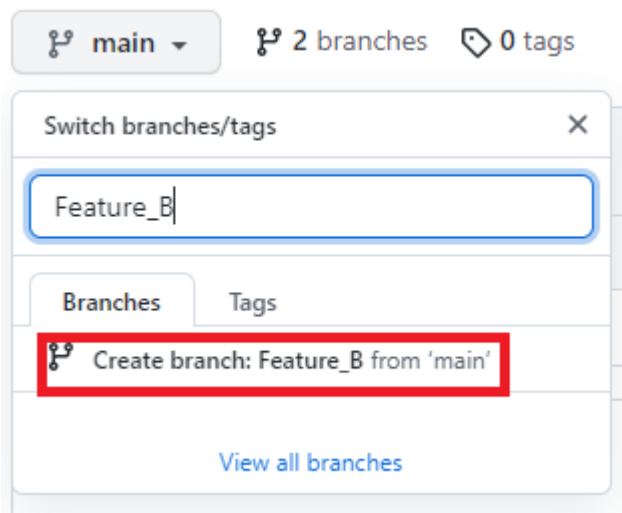
3. Select branch from which you want to make a copy from. I selected the main branch.



4. Start typing the new branch name in the textbox.



5. An option to create a branch will appear, select that option and the branch will be created.



From Command Prompt:

1. `git branch` command shows the branch which you cloned into the local repository.

```
C:\Users\Sai Sucharitha Arava\OneDrive\Documents\trial>cd Trial_repo
C:\Users\Sai Sucharitha Arava\OneDrive\Documents\trial\Trial_repo>git branch
* main
```

2. `git branch new_branch_name` creates new branch. `git checkout new_branch_name` changes local working directory.

```
C:\Users\Sai Sucharitha Arava\OneDrive\Documents\trial\Trial_repo>git branch Feature_C
C:\Users\Sai Sucharitha Arava\OneDrive\Documents\trial\Trial_repo>git checkout Feature_C
Switched to branch 'Feature_C'
```

3. `Git checkout -b new_branch_name` is shorthand command for above both commands.

Push code to github:

1. *git status* shows the list of tracked, untracked files and changes.

```
C:\Users\Sai Sucharitha Arava\OneDrive\Documents\trial\Trial_repo>git status
On branch Feature_C
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:   .gitignore
    modified:   README.md

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    INFT6303_TeamD_Project.sln
    INFT6303_TeamD_Project/
    packages/

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

2. *git add filename* to add a new or changed file in your working directory to the Git staging area. *git add .* or *git add ** to add all files to the staging area.

```
C:\Users\Sai Sucharitha Arava\OneDrive\Documents\trial\Trial_repo>git add .
```

3. *git status* now you can see all files are in green color and added to the staging area.

```
C:\Users\Sai Sucharitha Arava\OneDrive\Documents\trial\Trial_repo>git status
On branch Feature_C
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    modified:   .gitignore
    new file:   INFT6303_TeamD_Project.sln
    new file:   INFT6303_TeamD_Project/App_Data/Feedback.mdf
    new file:   INFT6303_TeamD_Project/App_Data/Feedback_log.ldf
    new file:   INFT6303_TeamD_Project/Global.asax
    new file:   INFT6303_TeamD_Project/Global.asax.cs
    new file:   INFT6303_TeamD_Project/INFT6303_TeamD_Project.csproj
    new file:   INFT6303_TeamD_Project/Login_page.aspx
    new file:   INFT6303_TeamD_Project/Login_page.aspx.cs
    new file:   INFT6303_TeamD_Project/Login_page.aspx.designer.cs
    new file:   INFT6303_TeamD_Project/Properties/AssemblyInfo.cs
    new file:   INFT6303_TeamD_Project/Web.Debug.config
```

4. *git commit -m "commit message"* Adding commits keep track of our progress and changes as we work. Git considers each commit change point or "save point". It is a point in the project you can go back to if you find a bug, or want to make a change.

```
C:\Users\Sai Sucharitha Arava\OneDrive\Documents\trial\Trial_repo>git commit -m "Initial commit"
[Feature_C 255bf21] Initial commit
 112 files changed, 3405 insertions(+), 29 deletions(-)
 create mode 100644 INFT6303_TeamD_Project.sln
```

5. *git push origin branch_name* to push the code to remote repository.

```
C:\Users\Sai Sucharitha Arava\OneDrive\Documents\trial\Trial_repo>git push origin Feature_C
Enumerating objects: 126, done.
Counting objects: 100% (126/126), done.
Delta compression using up to 4 threads
Compressing objects: 100% (121/121), done.
Writing objects: 100% (123/123), 26.26 MiB | 2.20 MiB/s, done.
Total 123 (delta 44), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (44/44), done.
remote:
remote: Create a pull request for 'Feature_C' on GitHub by visiting:
remote:   https://github.com/suchi98042/Trial_repo/pull/new/Feature_C
remote:
To https://github.com/suchi98042/Trial_repo.git
 * [new branch]      Feature_C -> Feature_C
```

Pull code from git:

git pull <remote> branch_name used to fetch and download content from a remote repository and immediately update the local repository to match that content.

```
C:\Users\Sai Sucharitha Arava\OneDrive\Documents\trial\Trial_repo>git pull origin Feature_C
From https://github.com/suchi98042/Trial_repo
 * branch                Feature_C  -> FETCH_HEAD
Updating 255bf21..8092375
Fast-forward
 INFT6303_TeamD_Project/Login_page.aspx | 2 +-
 1 file changed, 1 insertion(+), 1 deletion(-)
```

Merging two branches:

It's preferred to change/switch to the master branch before any branch needs to be merged with it. This will merge the specified branch with our master branch.

git merge <branch_name>

Deleting a branch:

git branch -d <branch_name>