



# Installation setup for Git

## Prerequisites for Installing Git and Git Bash

Before installing Git and Git Bash, you must have these in your system:

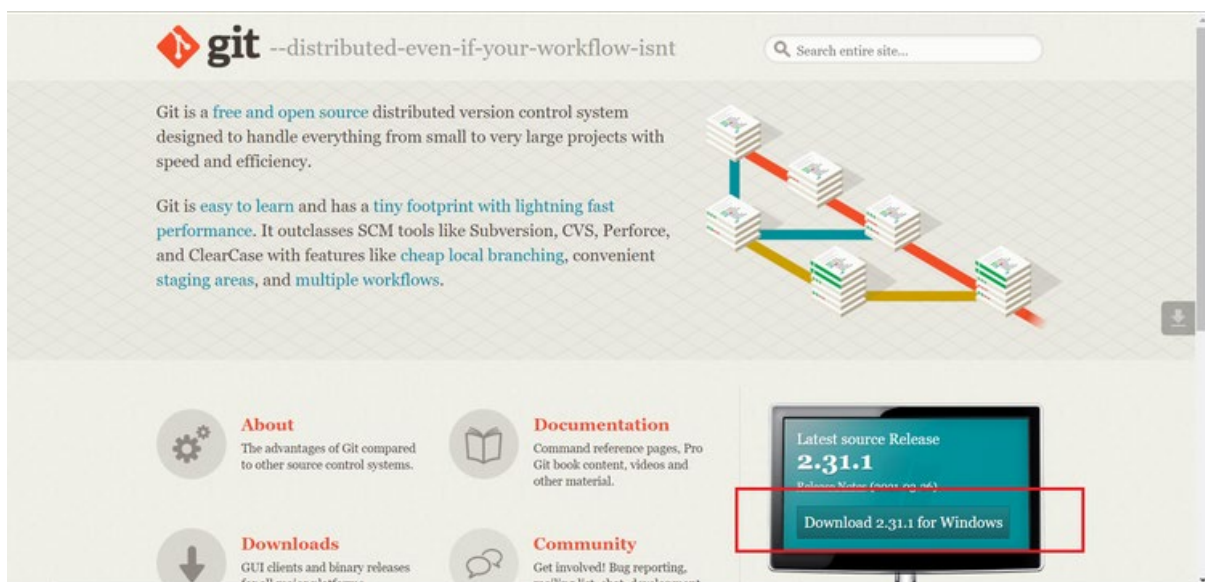
- Administrator privileges
- Command-Line access
- Coding text editor
- GitHub Username and Password

## Download and Install Git for Windows

You can download Git and Git Bash on Windows by following these simple steps:

### Step 1: Go to the Official Git Website

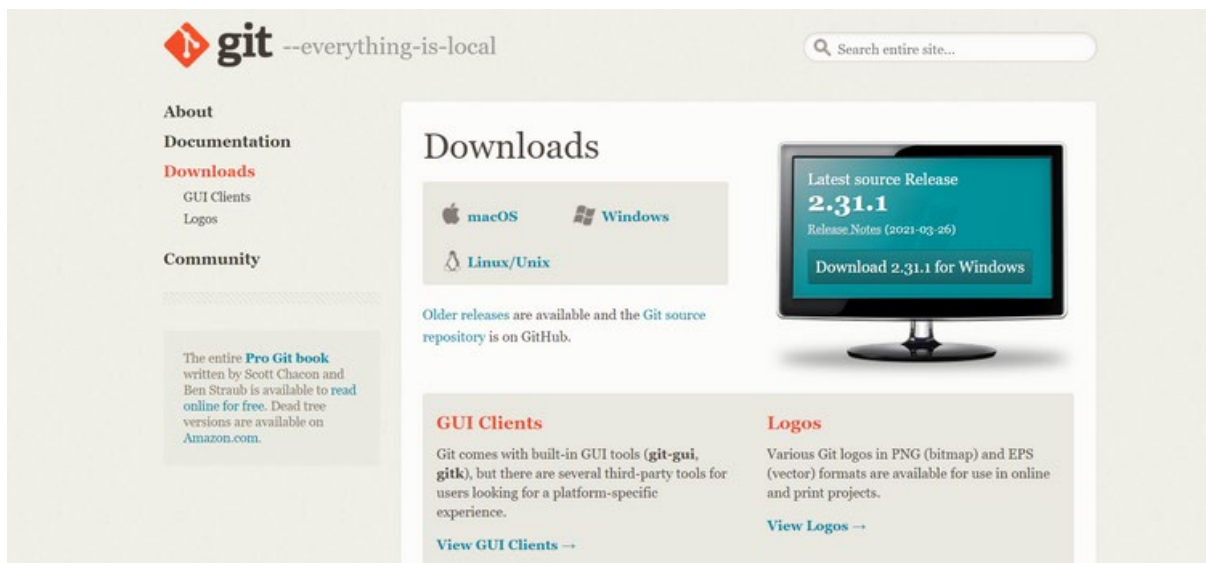
Visit the official website for [Git](https://git-scm.com/) and click the **Download [version] for Windows** button. The download will be started automatically after you click the button.



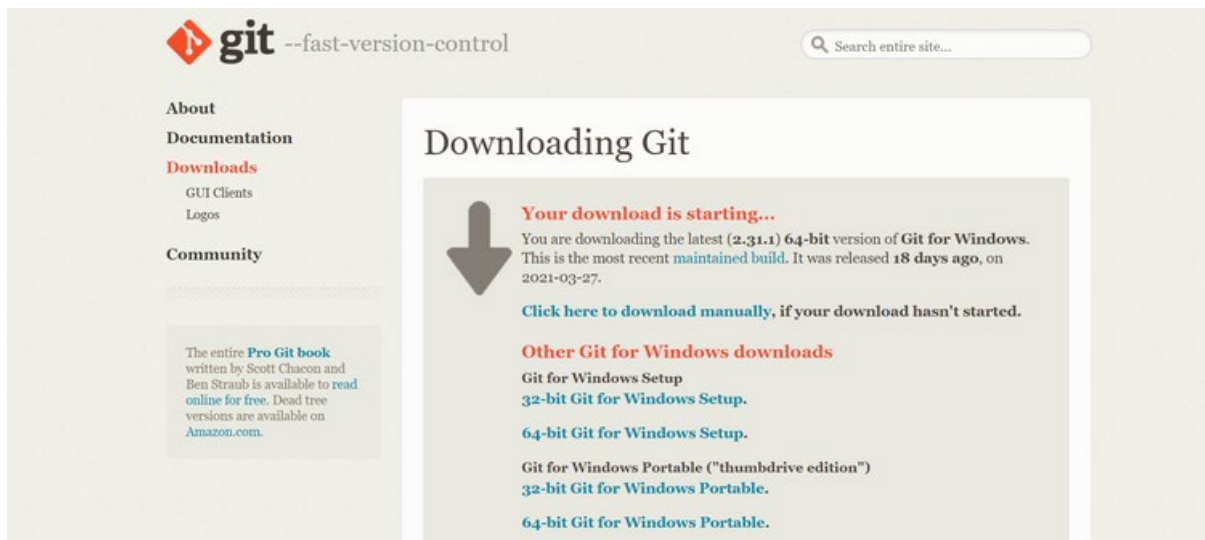
Alternatively, you can visit the downloads page of the official Git Website by clicking the **Downloads** button.



Click on the **Windows** button to start the download automatically.



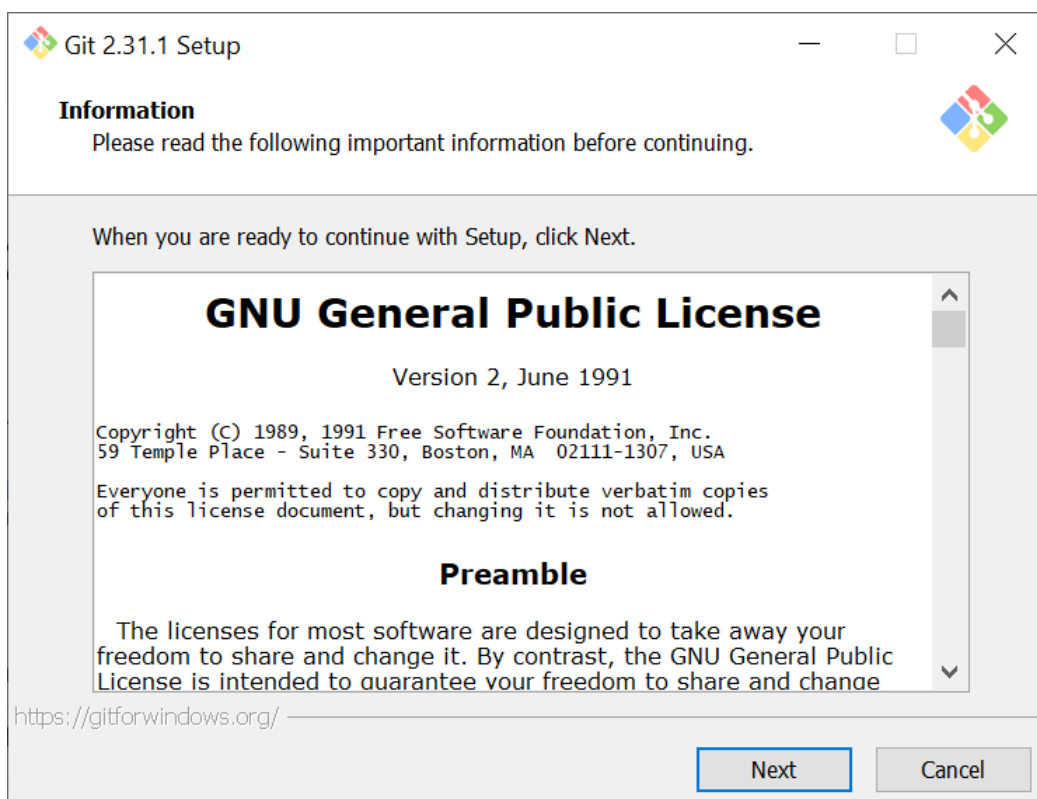
If the download doesn't start automatically, click on the **Click here to download manually** button.



## Step 2: Run the Downloaded File

After you've downloaded the executable file, click on it to run the installer. A pop-up window asking permission to make changes to the device will be displayed. Click on **Yes** to accept the request. After that, the Git Setup window will be opened.

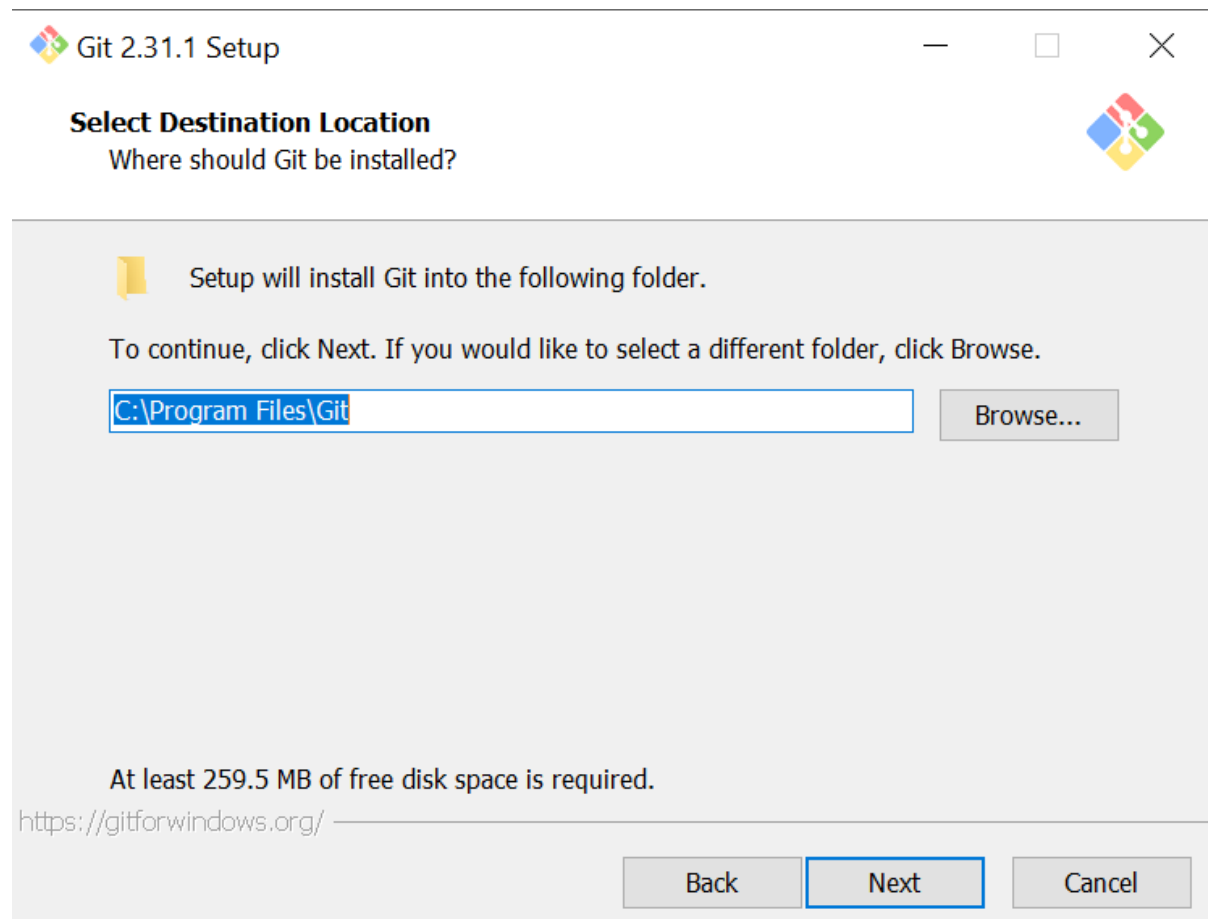
Carefully read the License and then when you're ready, click the **Next** button.





### Step 3: Select Destination Location

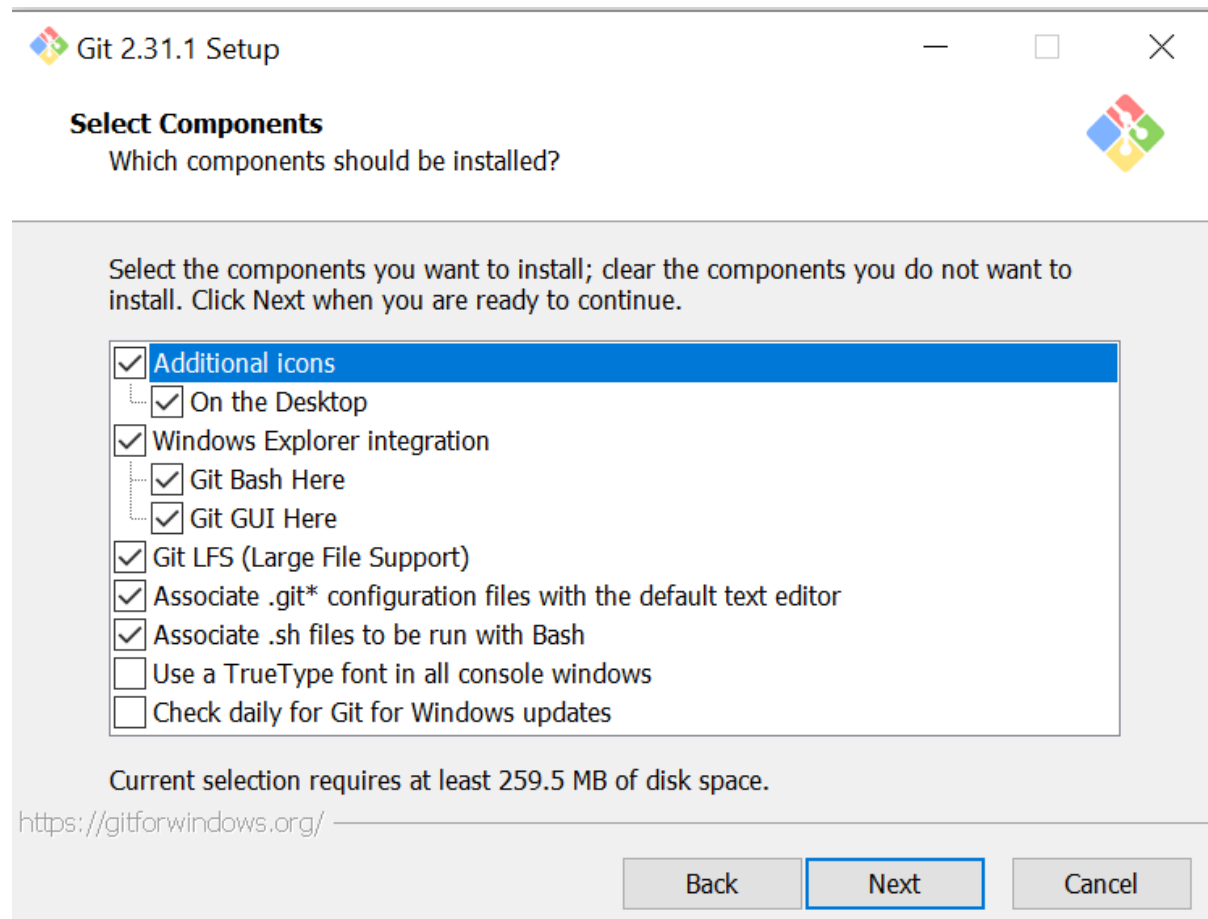
Click on the **Browse...** button to select the destination location where you want to install Git. By default, it'll install to **C:\Program Files\Git**. Click on the **Next** button after you've chosen your destination location.





## Step 4: Select Components

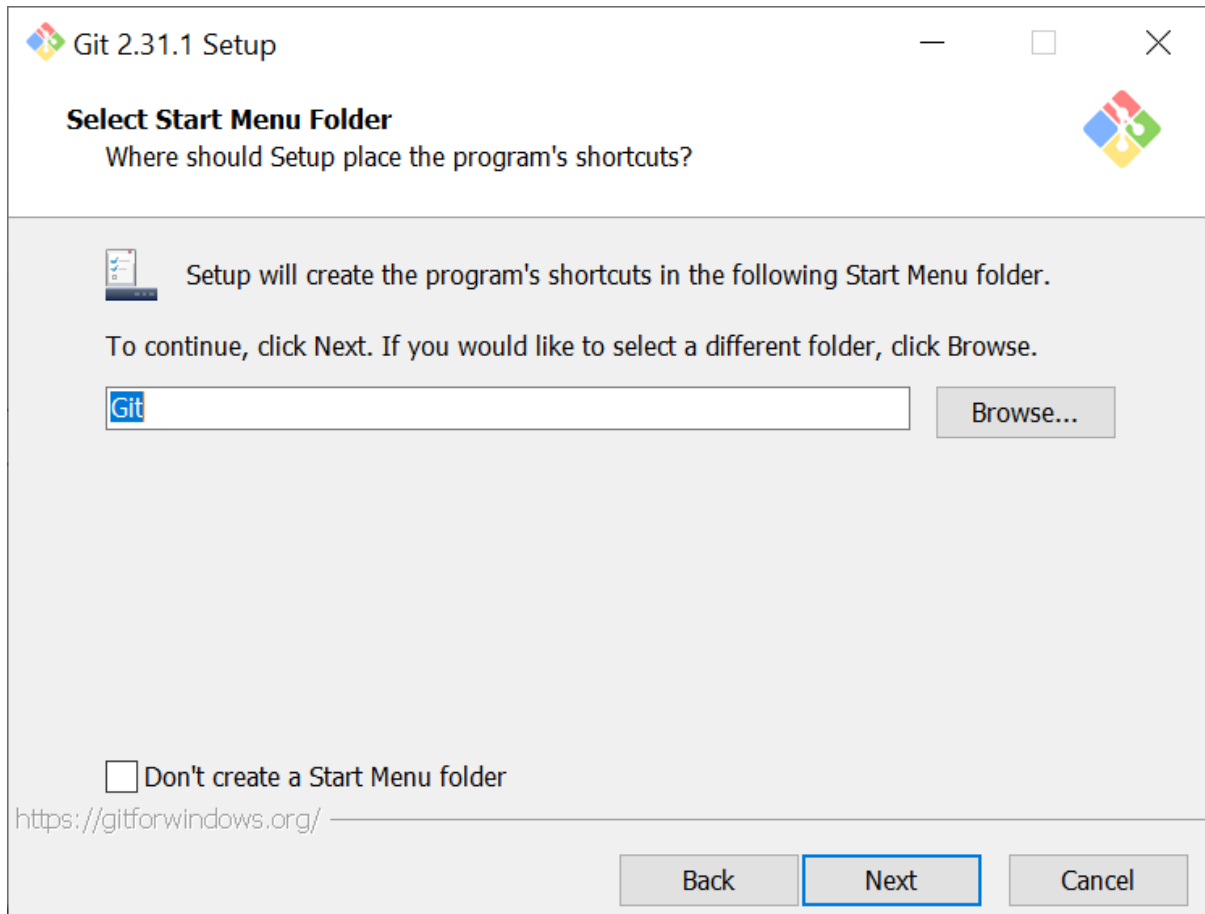
Click on the checkbox to install additional components such as a desktop icon. But if you prefer, you may proceed with default settings. Also, make sure that the "Git Bash Here" checkbox is checked. Hit the **Next** button to move to the next step.





## Step 5: Select Start Menu Folder

If you'd like, you may change the start menu folder name. It's headache-free to keep it as it is, however. Click **Next** to proceed with further steps.



## Step 6: Choose the Default Editor to be Used by Git

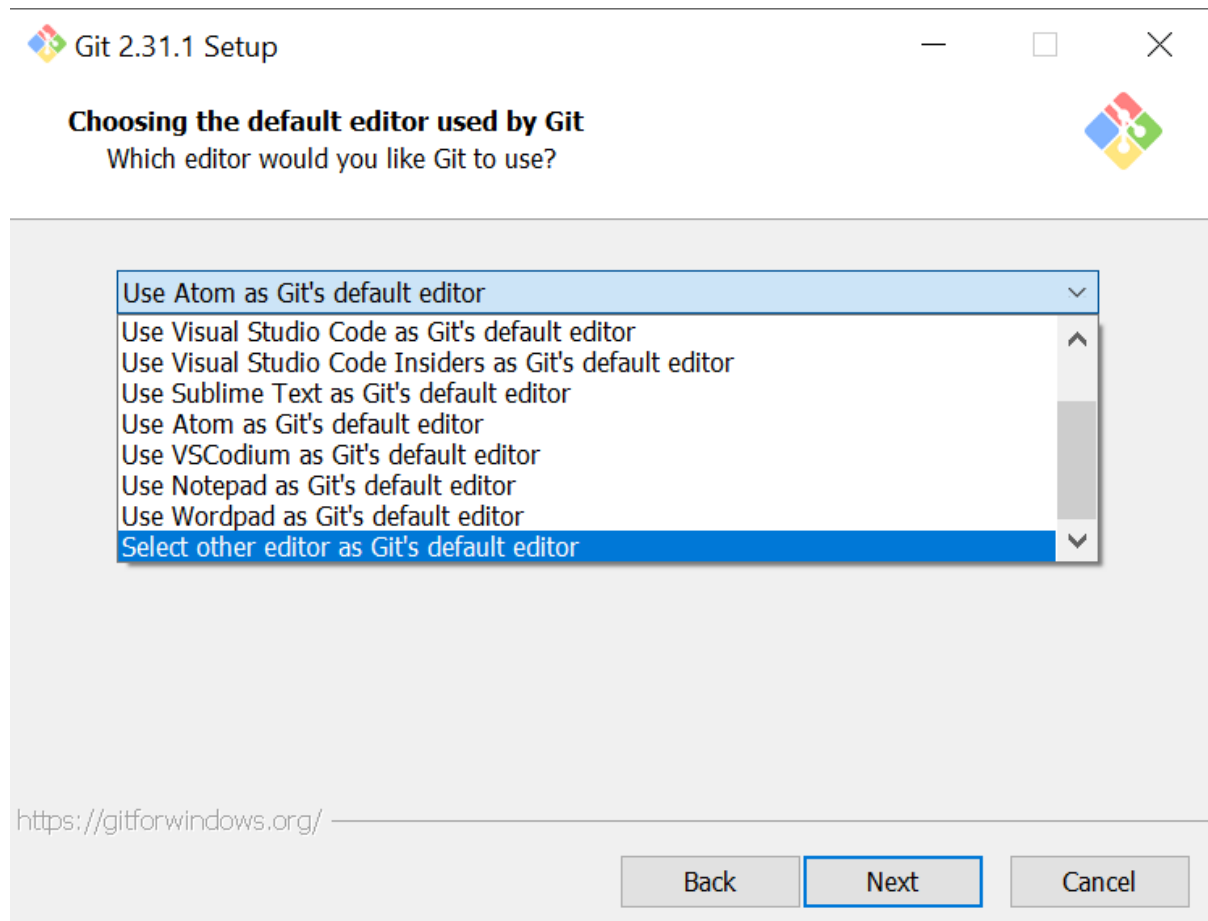
Choose the default text editor you want Git to use among various available options like Vim, Atom, Visual Studio Code, Sublime Text, Notepad, Wordpad, etc.

It's recommended to use **Visual Studio Code or Atom** as a default editor as they are the most widely used editors and have various cool features. Also, Vim is not recommended for beginners because it has a steep learning curve.

PS: (I have chosen Vim and this doesn't affect too much. So, don't worry, just choose the one that suits you the most)



Click on the **Next** button to proceed further.

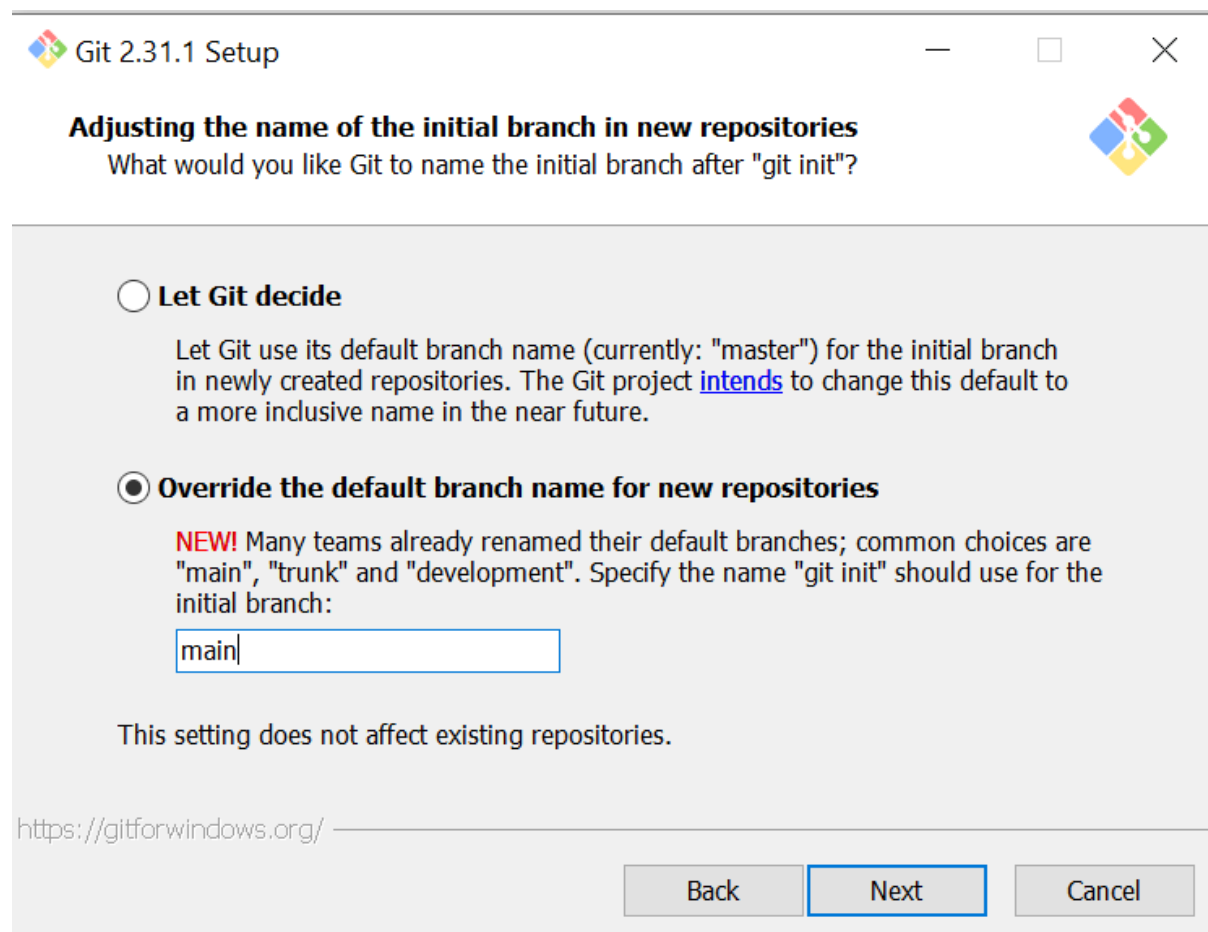


## Step 7: Adjusting the Name of the Initial Branch in New Repositories

It's recommended to select **Override the default branch name for new repositories** and use **main** as the default initial branch name.

The "git init" command will use the same initial branch name while initializing repositories. You can also use any other initial branch names like "default", "primary", "develop", "stable", "release", etc. It completely depends on what suits you best.

Finally, click the **Next** button to proceed after specifying the branch name.



The image shows the 'Git 2.31.1 Setup' window. The title bar says 'Git 2.31.1 Setup'. The main heading is 'Adjusting the name of the initial branch in new repositories'. Below it is the question 'What would you like Git to name the initial branch after "git init"?'. There are two radio button options. The first is 'Let Git decide' with a description: 'Let Git use its default branch name (currently: "master") for the initial branch in newly created repositories. The Git project [intends](#) to change this default to a more inclusive name in the near future.' The second option is selected: 'Override the default branch name for new repositories'. Below this is a 'NEW!' note: 'Many teams already renamed their default branches; common choices are "main", "trunk" and "development". Specify the name "git init" should use for the initial branch:'. There is a text input field containing 'main'. Below the input field is the text 'This setting does not affect existing repositories.' At the bottom left is the URL 'https://gitforwindows.org/'. At the bottom right are three buttons: 'Back', 'Next' (which is highlighted with a blue border), and 'Cancel'.

Git 2.31.1 Setup

### Adjusting the name of the initial branch in new repositories

What would you like Git to name the initial branch after "git init"?

☐ **Let Git decide**

Let Git use its default branch name (currently: "master") for the initial branch in newly created repositories. The Git project [intends](#) to change this default to a more inclusive name in the near future.

☒ **Override the default branch name for new repositories**

**NEW!** Many teams already renamed their default branches; common choices are "main", "trunk" and "development". Specify the name "git init" should use for the initial branch:

This setting does not affect existing repositories.

<https://gitforwindows.org/>

Back Next Cancel

It used to be that "master" was used as the default initial branch name for GitHub repositories. But now it's changed to "main" as some people found "master" an offensive word. GitHub followed the Software Freedom Conservancy's suggestion and moved away from the term "master" when a Git repository is initialized.

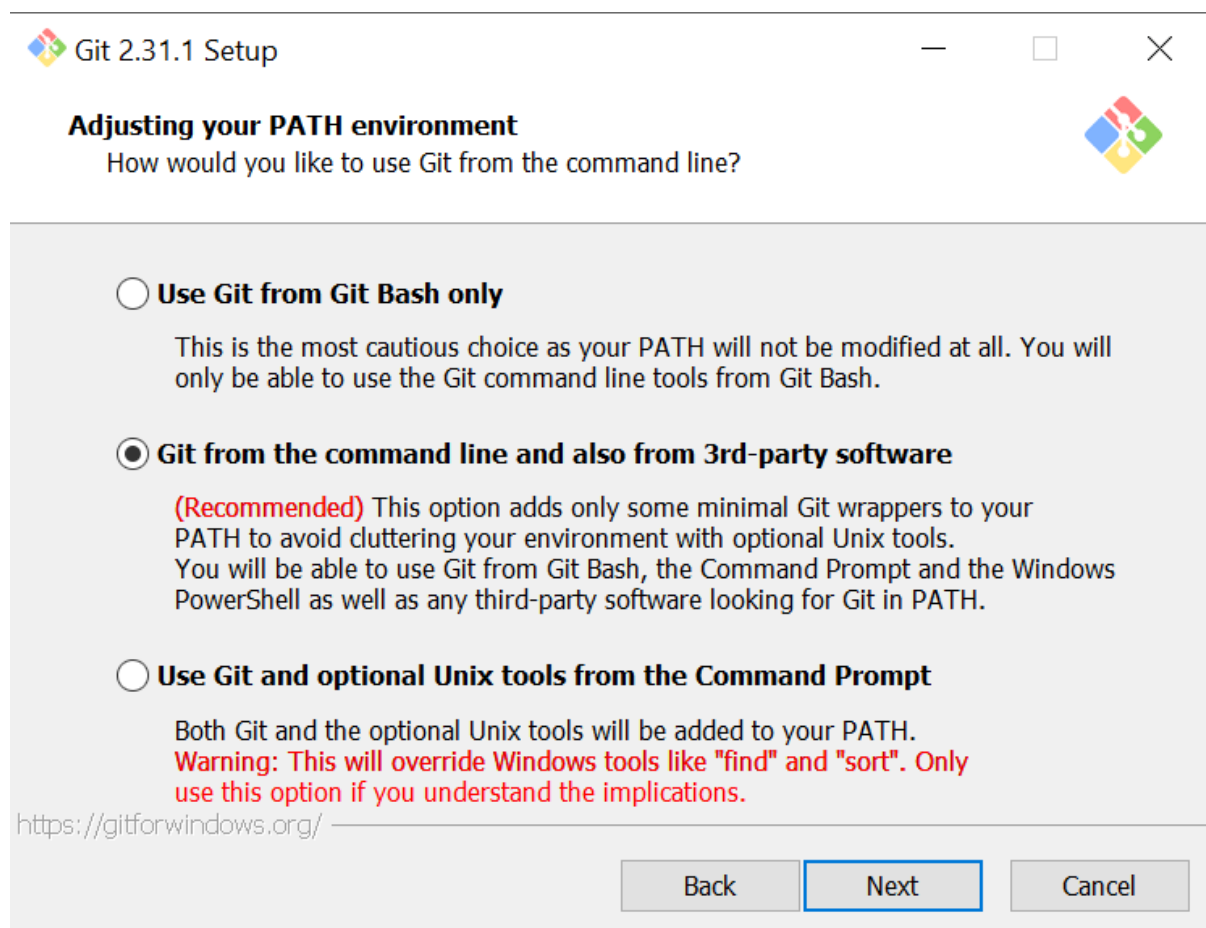




## Step 8: Adjust Your PATH Environment

Select the 2nd option **Git from the command line and also from 3rd-party software**. By selecting this option you will be able to use **Git** from the **Git Bash**, the Command Prompt, the Windows Powershell, or any other 3rd party software looking for Git in PATH.

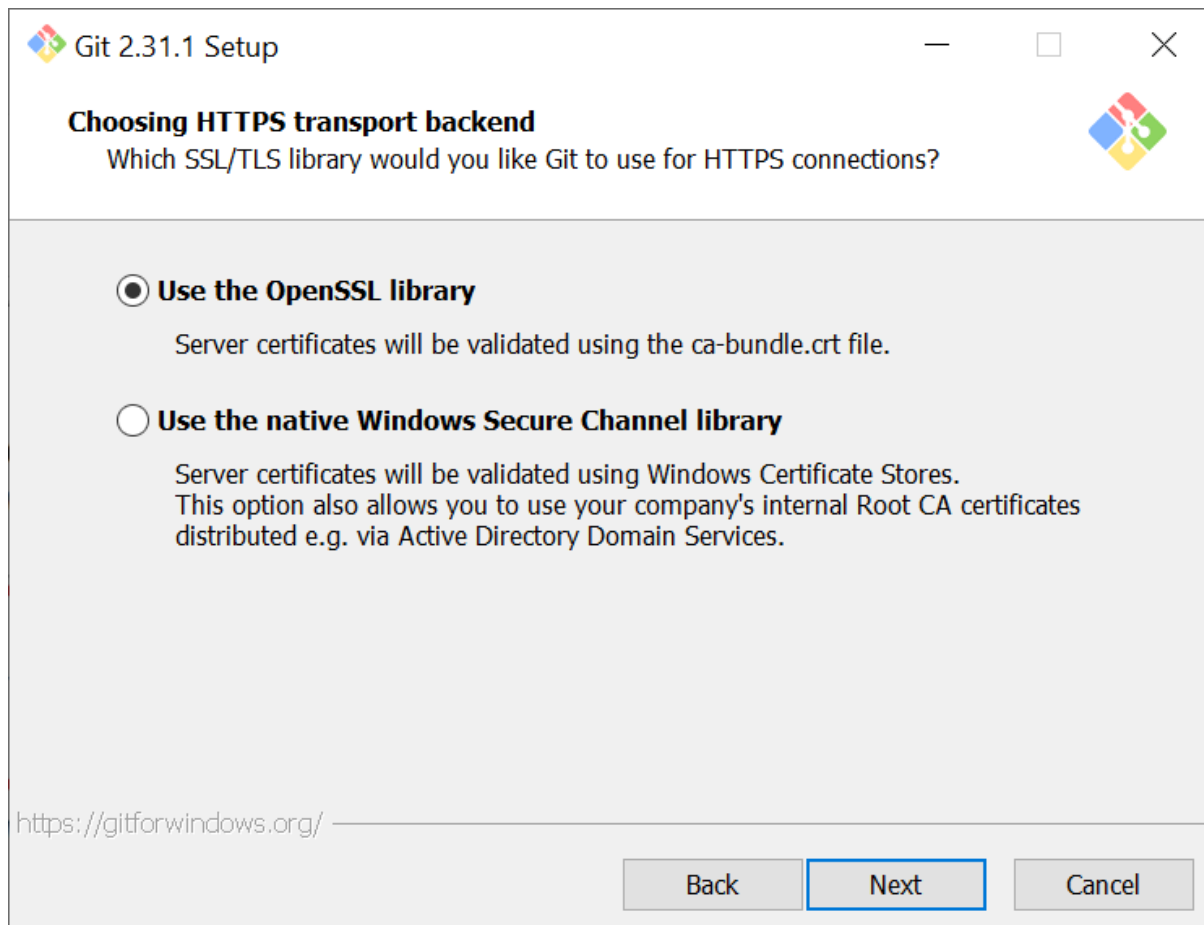
Hit the **Next** button to proceed.





## Step 9: Choosing HTTPS Transport Backend

Select the **Use the OpenSSL Library** option and click **Next**.





## Step 10: Configuring the Line Ending Conversions

Proceed with the by default selected option **Checkout Windows-style, commit Unix-style line endings** and then click **Next**.

A screenshot of the 'Git 2.31.1 Setup' window. The title bar says 'Git 2.31.1 Setup'. The main heading is 'Configuring the line ending conversions' with a subtitle 'How should Git treat line endings in text files?'. There are three radio button options. The first option, 'Checkout Windows-style, commit Unix-style line endings', is selected. The second option is 'Checkout as-is, commit Unix-style line endings'. The third option is 'Checkout as-is, commit as-is'. At the bottom, there are three buttons: 'Back', 'Next', and 'Cancel'. The 'Next' button is highlighted with a blue border. A URL 'https://gitforwindows.org/' is visible at the bottom left.

**Git 2.31.1 Setup**

**Configuring the line ending conversions**  
How should Git treat line endings in text files?

☒ **Checkout Windows-style, commit Unix-style line endings**  
Git will convert LF to CRLF when checking out text files. When committing text files, CRLF will be converted to LF. For cross-platform projects, this is the recommended setting on Windows ("core.autocrlf" is set to "true").

☐ **Checkout as-is, commit Unix-style line endings**  
Git will not perform any conversion when checking out text files. When committing text files, CRLF will be converted to LF. For cross-platform projects, this is the recommended setting on Unix ("core.autocrlf" is set to "input").

☐ **Checkout as-is, commit as-is**  
Git will not perform any conversions when checking out or committing text files. Choosing this option is not recommended for cross-platform projects ("core.autocrlf" is set to "false").

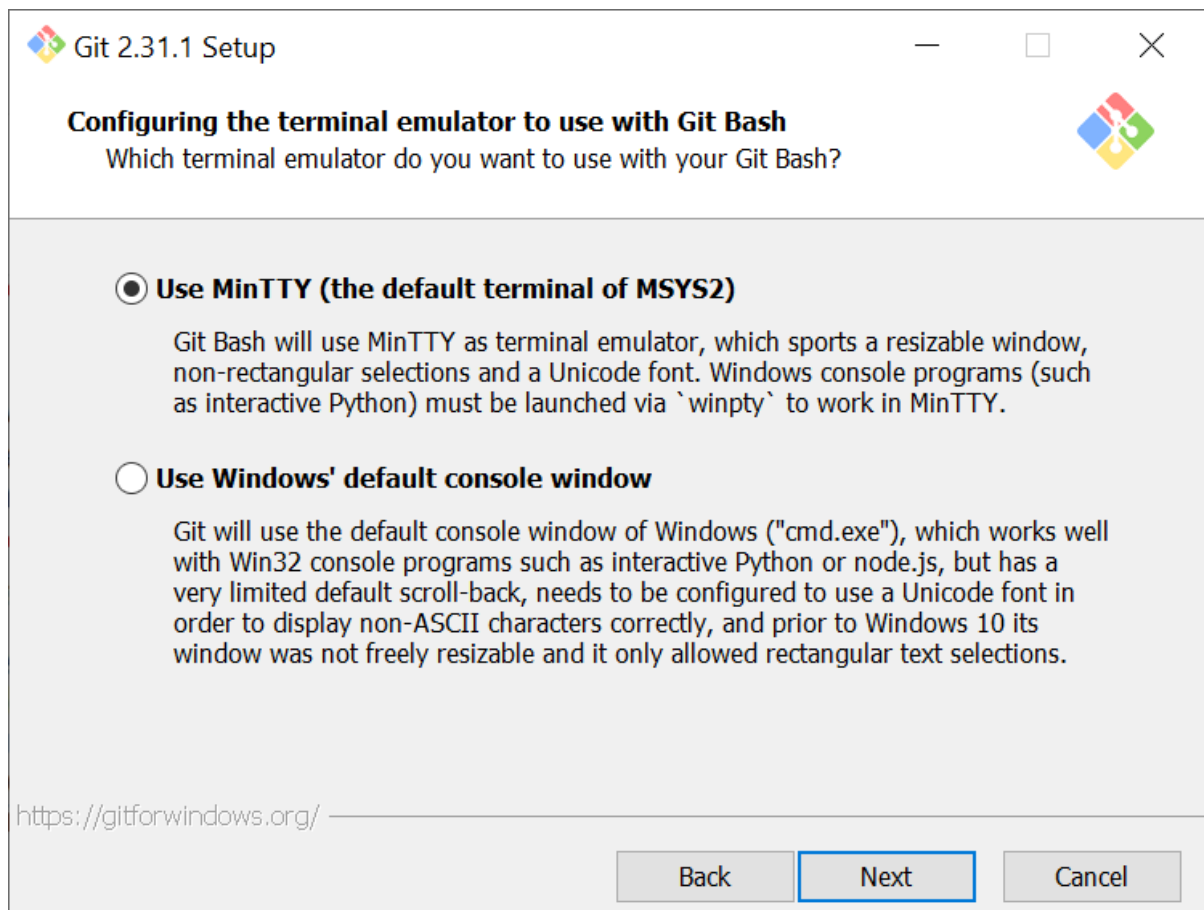
<https://gitforwindows.org/>

Back Next Cancel



## Step 11: Configuring the Terminal Emulator to Use With Git Bash

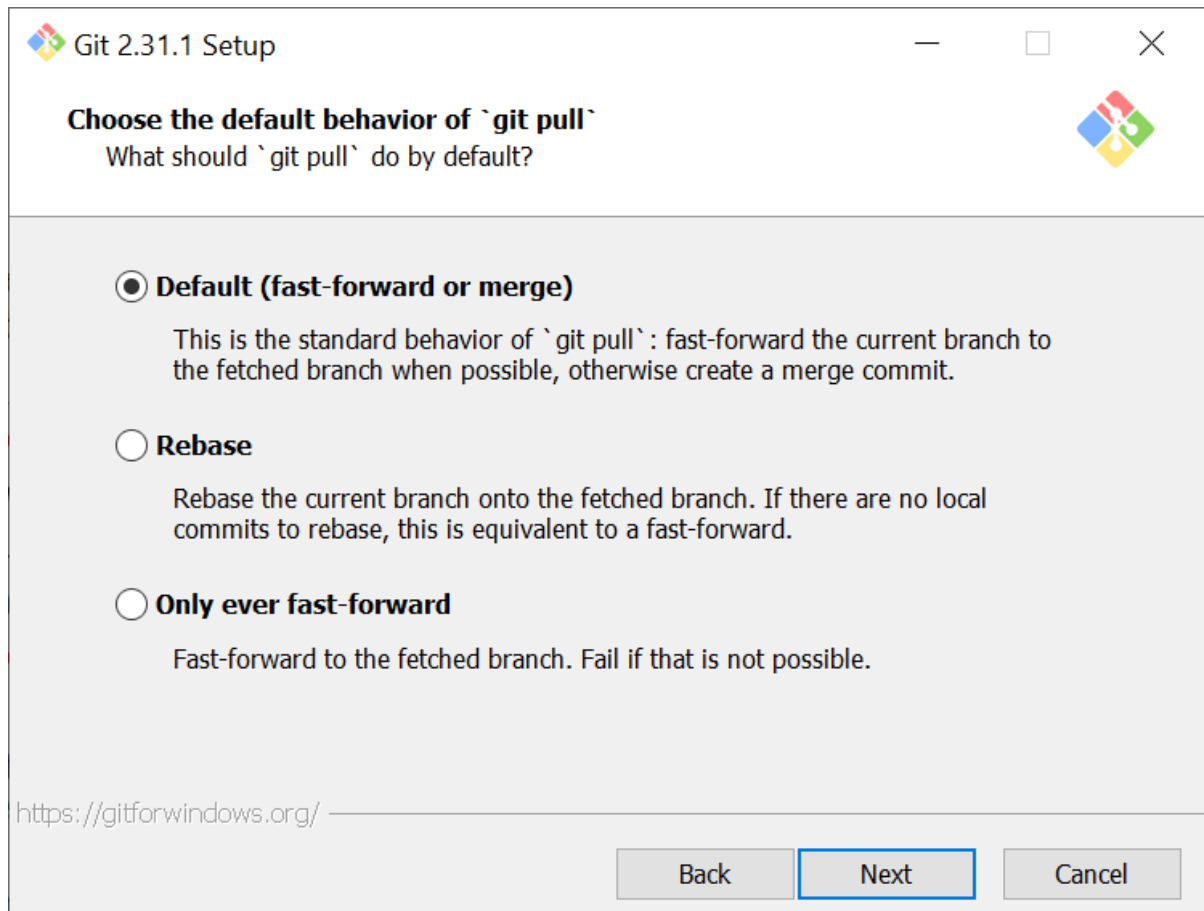
Again proceed with the default selected option **Use MinTTY (the default terminal of MSYS2)** and then click **Next**.





## Step 12: Choose the Default Behaviour of "git pull"

Select the first option **Default (fast-forward or merge)**. By selecting this option, when "git pull" is used, it'll fast-forward the current branch to the fetched branch. If it's not possible to do so, it'll create a merge commit.

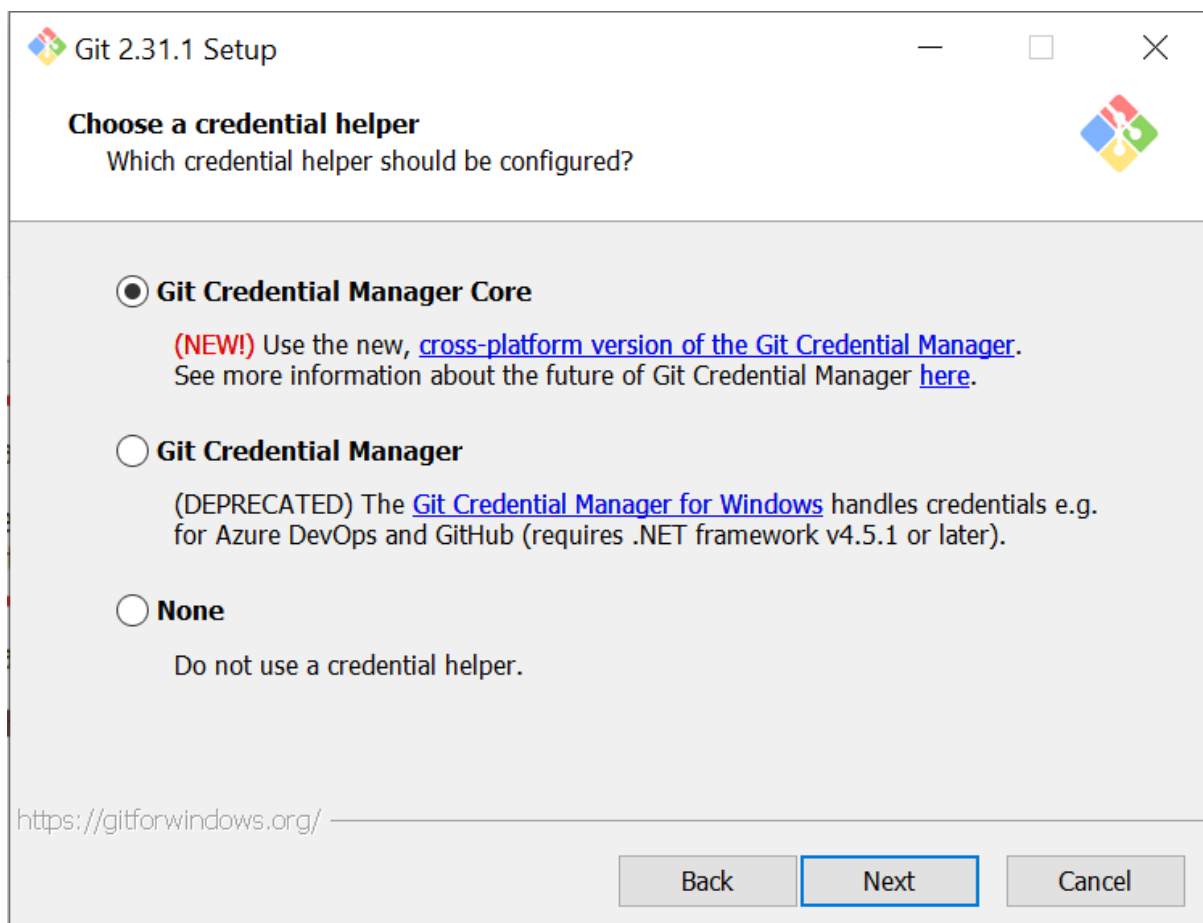




## Step 13: Choose a Credential Helper

We highly recommended selecting the first option, **Git Credential Manager Core**, as it provides a consistent authentication experience across all platforms.

After that, click the **Next** button to proceed.





## Step 14: Configuring Extra Options

Proceed further with the by default selected options and then click **Next**.

A screenshot of the "Git 2.31.1 Setup" window. The title bar says "Git 2.31.1 Setup" with standard window controls. The main content area is titled "Configuring extra options" with a subtitle "Which features would you like to enable?". There are two options: "Enable file system caching" which is checked, and "Enable symbolic links" which is unchecked. Below each option is a descriptive paragraph. At the bottom, there is a URL "https://gitforwindows.org/" and three buttons: "Back", "Next" (which is highlighted with a blue border), and "Cancel".

**Git 2.31.1 Setup**

**Configuring extra options**  
Which features would you like to enable?

☒ **Enable file system caching**  
File system data will be read in bulk and cached in memory for certain operations ("core.fscache" is set to "true"). This provides a significant performance boost.

☐ **Enable symbolic links**  
Enable [symbolic links](#) (requires the SeCreateSymbolicLink permission). Please note that existing repositories are unaffected by this setting.

<https://gitforwindows.org/>

Back Next Cancel

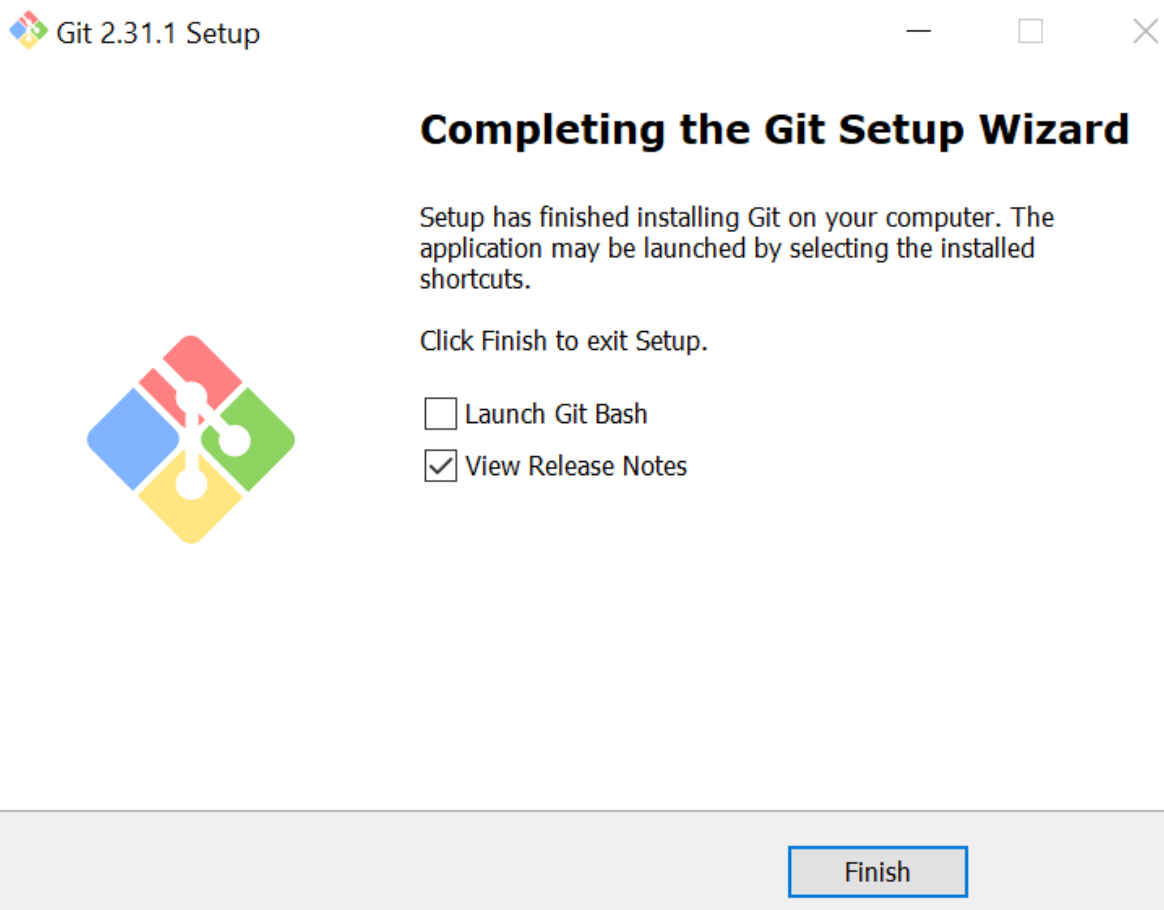


## Step 15: Configuring Experimental Options

If you want to enable some bleeding-edge features with this installation then you can select any of the available options. Finally, click the **Install** button.

## Step 16: Wait for Installation

Wait for few minutes as the Setup installs Git and Git Bash on your system. After the installation is complete, click **Finish** to exit Setup.



Now Git and Git Bash is successfully installed on your computer!





## Confirm That Git Is Successfully Installed

Open the Command Prompt and enter the following command to verify that Git was successfully installed.

```
git --version
```

A screenshot of a Windows Command Prompt window. The title bar says 'Select Command Prompt'. The window content shows the following text: 'Microsoft Windows [Version 10.0.19043.1288]', '(c) Microsoft Corporation. All rights reserved.', 'C:\Users\shaun>git --version', 'git version 2.30.0.windows.1', and 'C:\Users\shaun>'. The cursor is at the end of the last line.

```
Microsoft Windows [Version 10.0.19043.1288]
(c) Microsoft Corporation. All rights reserved.

C:\Users\shaun>git --version
git version 2.30.0.windows.1

C:\Users\shaun>
```

## Configure Git

The next step is to configure Git by adding your credentials to the system. This is important as it helps keep track of which user is committing changes to a project.

Open the terminal and configure your GitHub username:

```
git config --global user.name "your_github_username"
```

Then, add your email:

```
git config --global user.email "your_email@github.com"
```



Once done, you can confirm that the information is set by running:

```
git config --list
```

Output:

```
User.name = your_github_username  
User.email = your_email@github.com
```

Congratulations, you have finish git setup, and you are ready to go.

See you in the workshop 😊

For Mac OS user, please refer this [link](#)

**Shaun Liew,**

**Microsoft Learn Student Ambassador,**

**Email: [Shaun.Liew@studentambassadors.com](mailto:Shaun.Liew@studentambassadors.com)**