Git set up in Windows & Mac

Git Set Up at Windows:

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

Git is a version control system that allows developers to track a project and actively contribute without interfering in each other's work.

It supports collaboration within a project and helps prevent miscommunication or code clashing between team members. The system tracks and saves snapshots of an evolving project, essentially keeping a history of the development.

Users who install the software on their machines can communicate with each other through the system. An even better solution is collaborating over a centralized source (for example, GitHub) where developers can push and pull changes on the cloud.

Pre-requisites

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

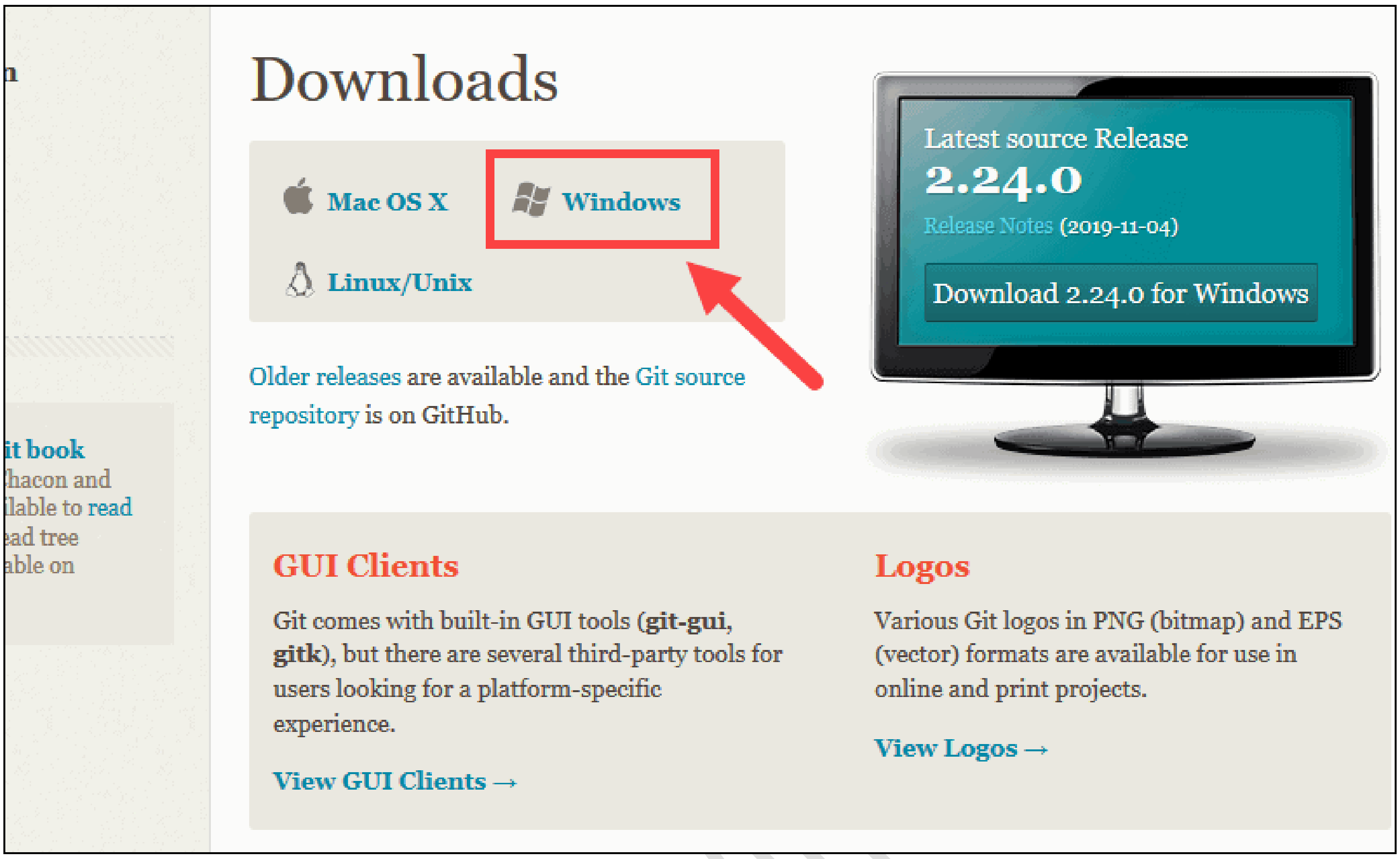
Steps For Installing Git for Windows

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.

Note: If you are new to Git, refer to our post How Does Git Work to learn more about Git workflow and Git functions.

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.



Downloads

Mac OS X | Windows

Linux/Unix

Older releases are available and the Git source repository is on GitHub.

Latest source Release
2.24.0
Release Notes (2019-11-04)
Download 2.24.0 for Windows

GUI Clients

Git comes with built-in GUI tools (**git-gui**, **gitk**), but there are several third-party tools for users looking for a platform-specific experience.

View GUI Clients →

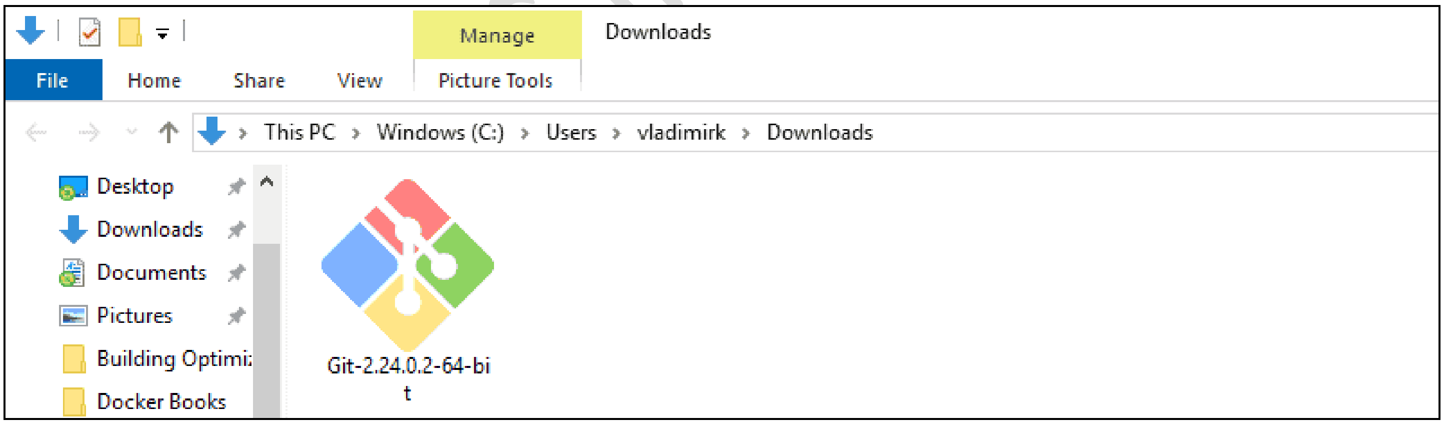
Logos

Various Git logos in PNG (bitmap) and EPS (vector) formats are available for use in online and print projects.

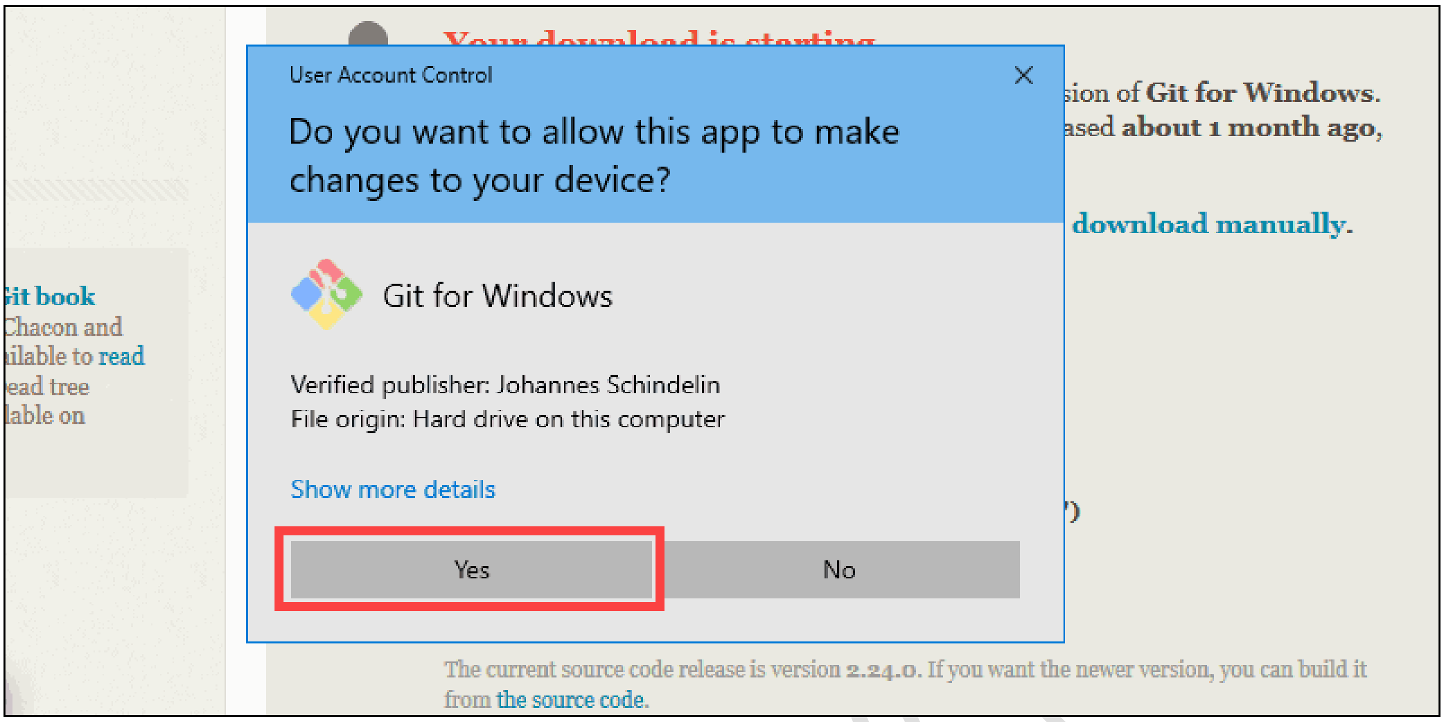
View Logos →

Extract and Launch Git Installer

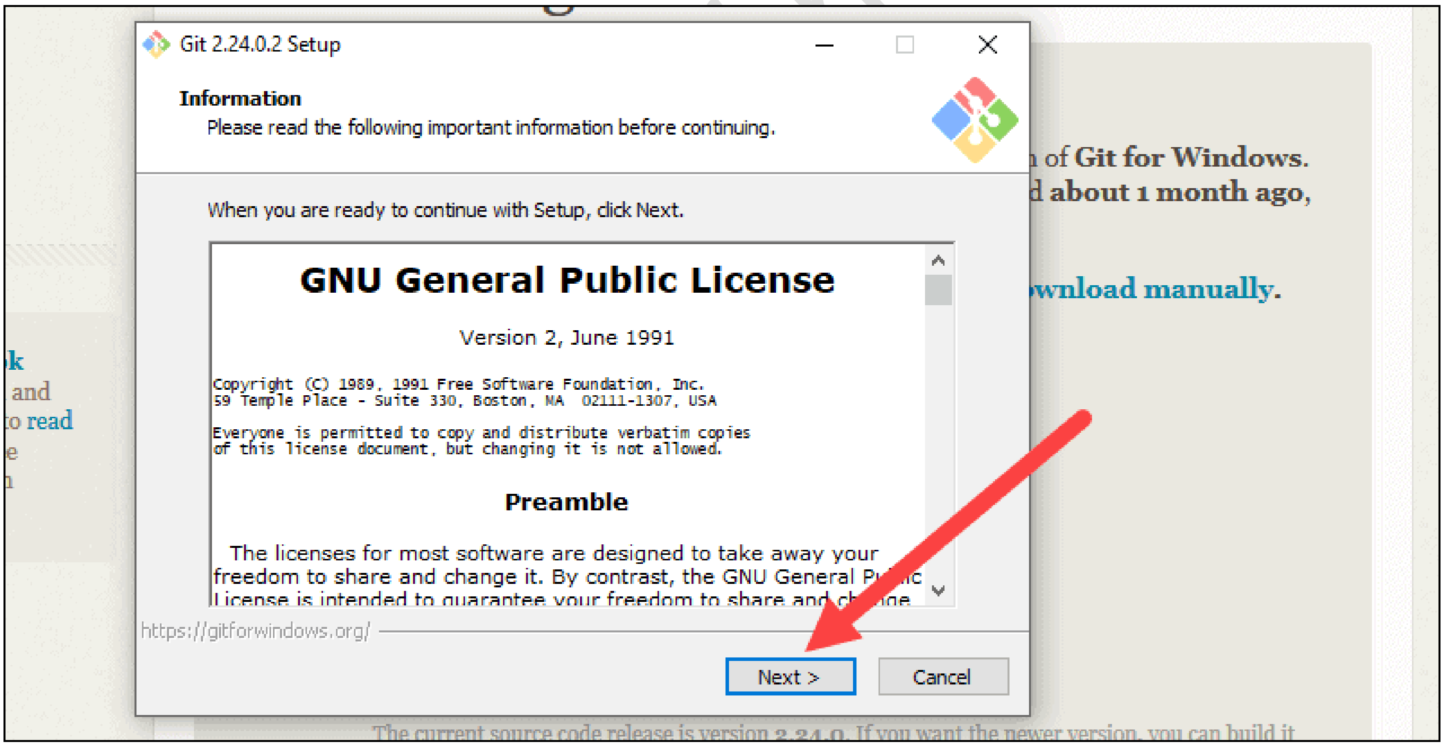
3. Browse to the download location (or use the download shortcut in your browser). Double-click the file to extract and launch the installer.



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


User Account Control
Do you want to allow this app to make changes to your device?
Git for Windows
Verified publisher: Johannes Schindelin
File origin: Hard drive on this computer
Show more details
Yes
No

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


Git 2.24.0.2 Setup
Information
Please read the following important information before continuing.
When you are ready to continue with Setup, click Next.
https://gitforwindows.org/
Next >
Cancel

GNU General Public License

Version 2, June 1991

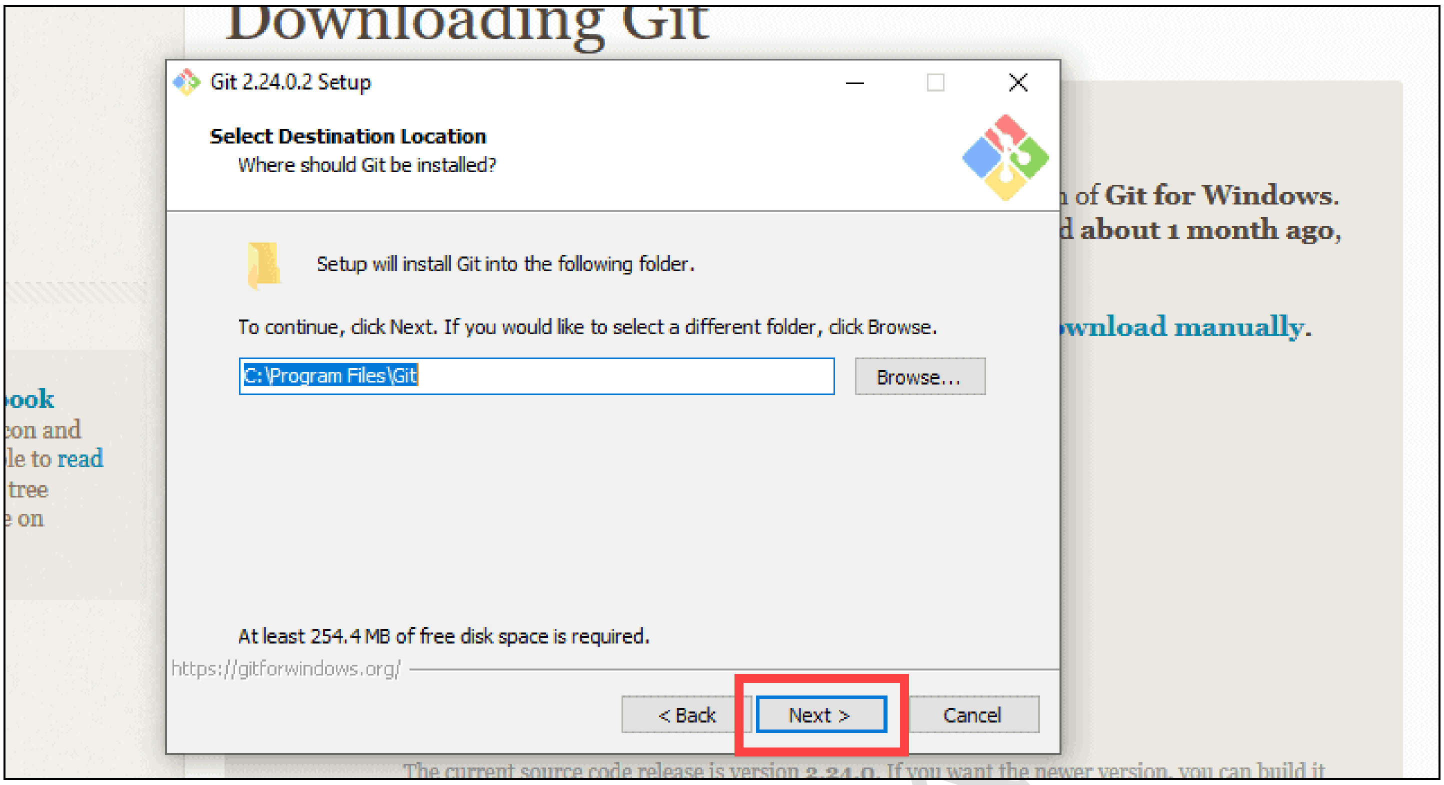
```
Copyright (C) 1989, 1991 Free Software Foundation, Inc.
59 Temple Place - Suite 330, Boston, MA  02111-1307, USA

Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.
```

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change

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


Git 2.24.0.2 Setup

Select Destination Location

Where should Git be installed?

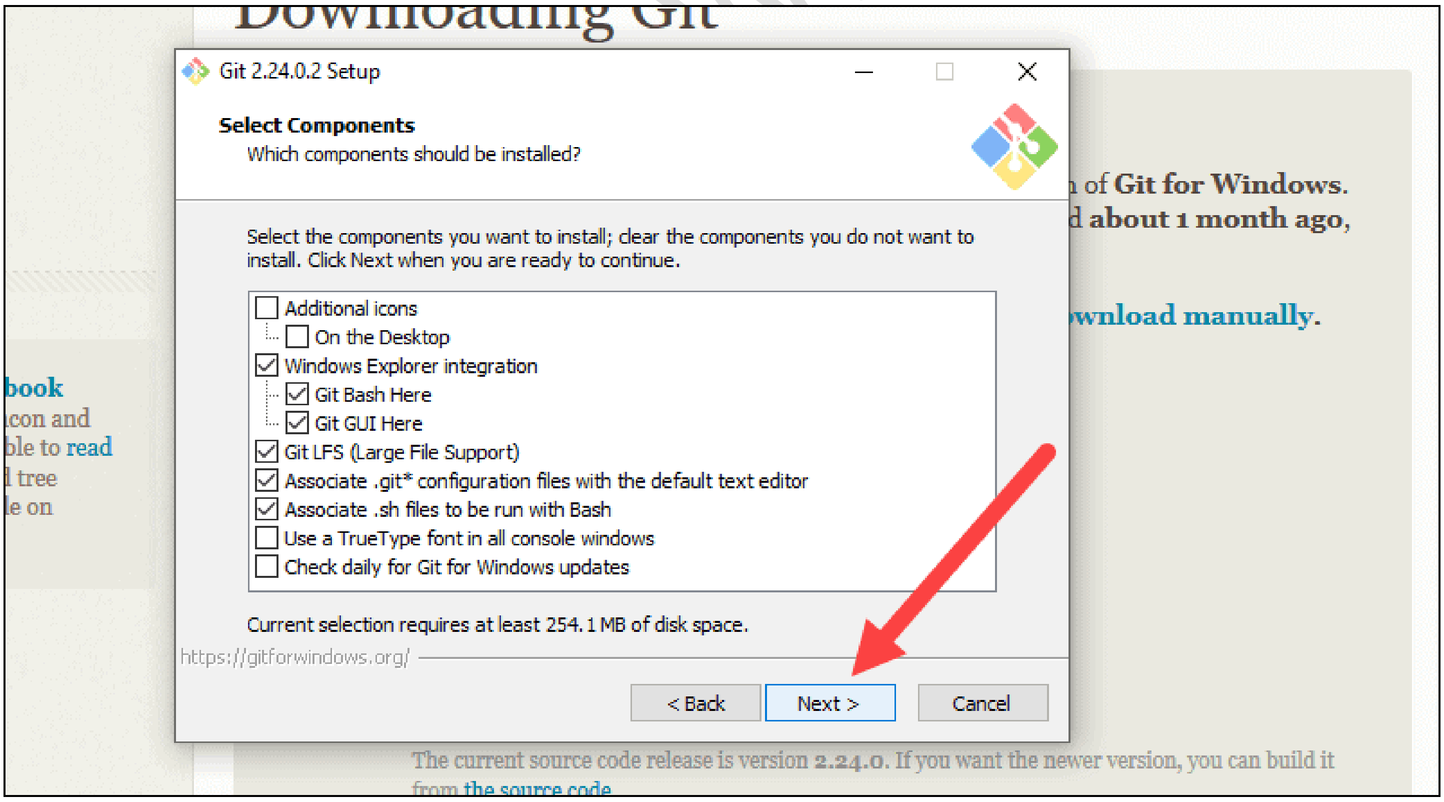
Setup will install Git into the following folder.

To continue, click Next. If you would like to select a different folder, click Browse.

C:\Program Files\Git

At least 254.4 MB of free disk space is required.

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


Git 2.24.0.2 Setup

Select Components

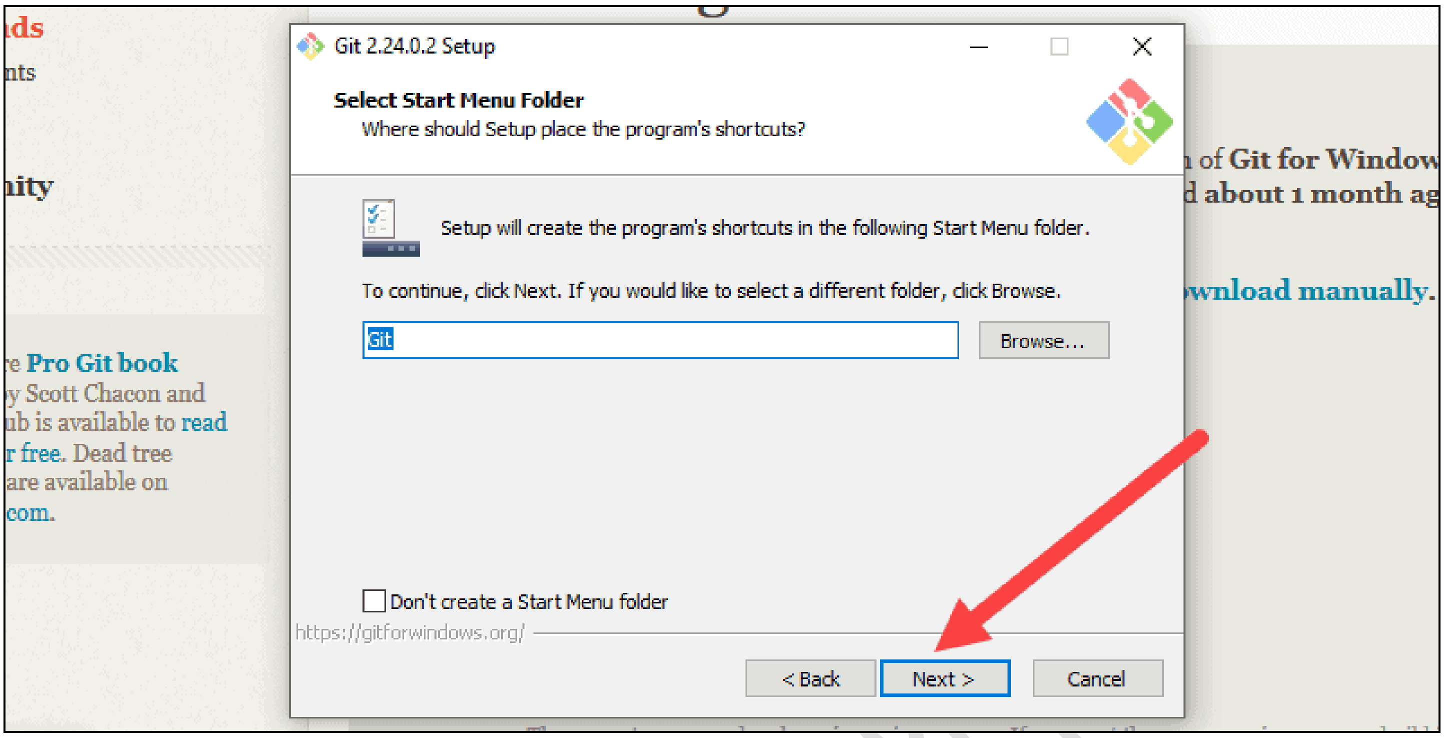
Which components should be installed?

Select the components you want to install; clear the components you do not want to install. Click Next when you are ready to continue.

- [ ] Additional icons
  - [ ] On the Desktop
- [x] Windows Explorer integration
  - [x] Git Bash Here
  - [x] Git GUI Here
- [x] Git LFS (Large File Support)
- [x] Associate .git* configuration files with the default text editor
- [x] Associate .sh files to be run with Bash
- [ ] Use a TrueType font in all console windows
- [ ] Check daily for Git for Windows updates

Current selection requires at least 254.1 MB of disk space.

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



Git 2.24.0.2 Setup

Select Start Menu Folder

Where should Setup place the program's shortcuts?

Setup will create the program's shortcuts in the following Start Menu folder.

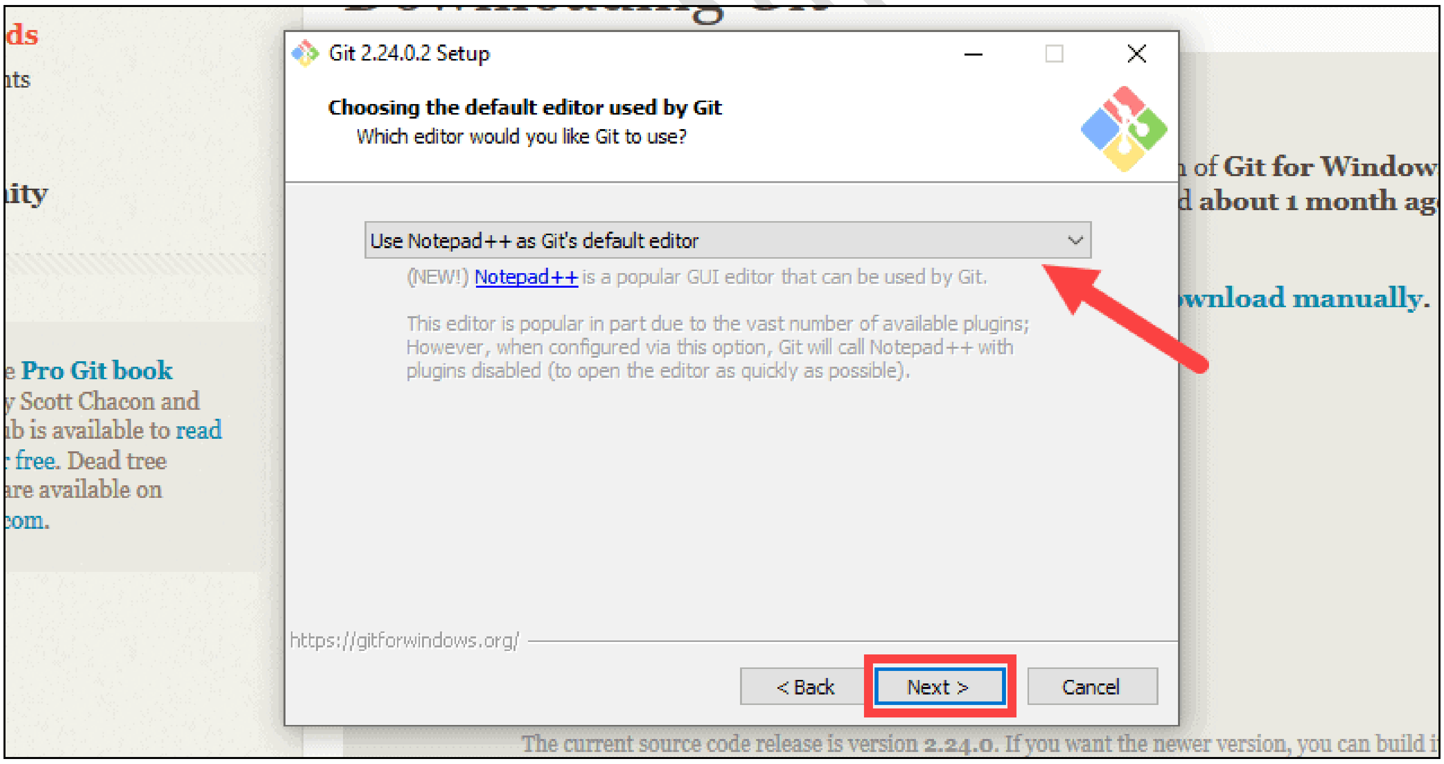
To continue, click Next. If you would like to select a different folder, click Browse.

Git

☐ Don't create a Start Menu folder

https://gitforwindows.org/

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**.



Git 2.24.0.2 Setup

Choosing the default editor used by Git

Which editor would you like Git to use?

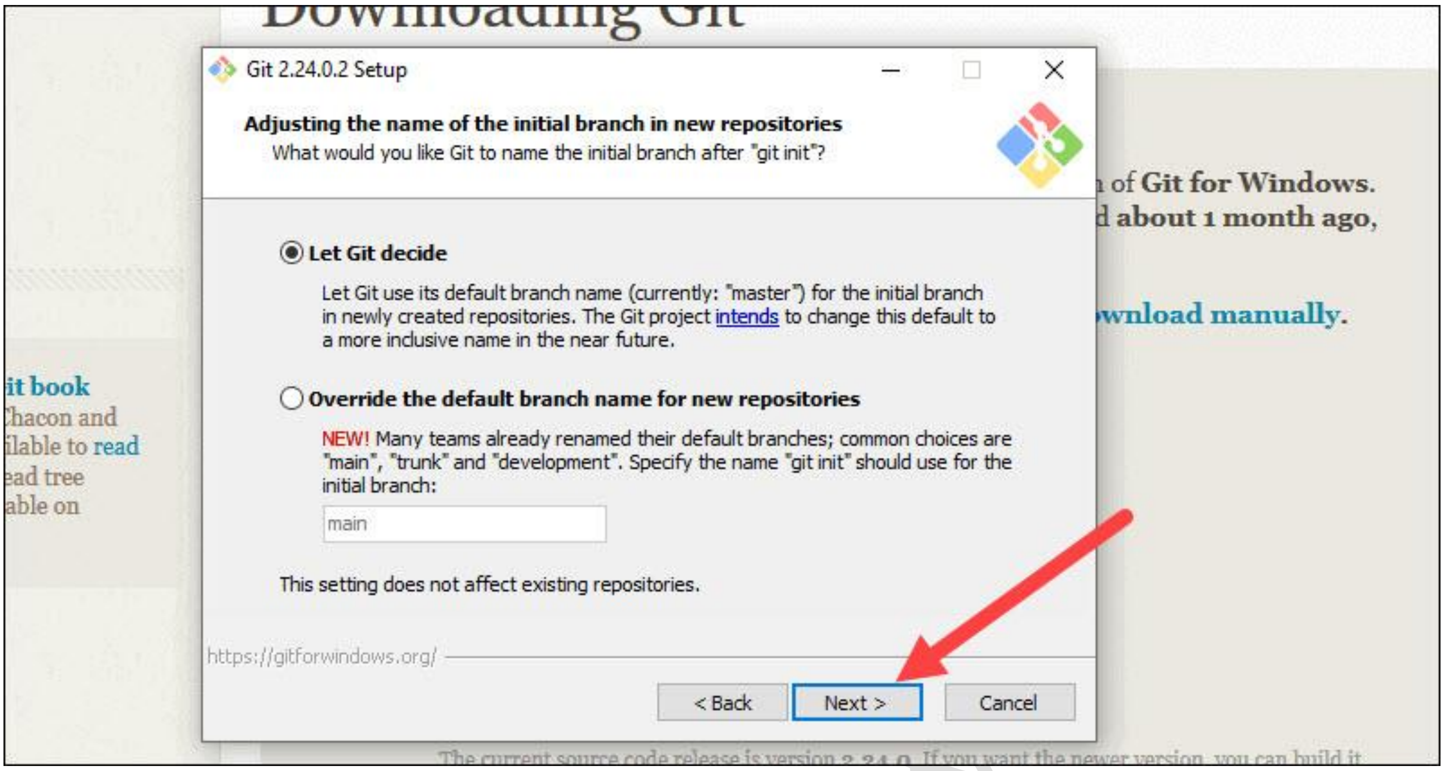
Use Notepad++ as Git's default editor

(NEW!) Notepad++ is a popular GUI editor that can be used by Git.

This editor is popular in part due to the vast number of available plugins; However, when configured via this option, Git will call Notepad++ with plugins disabled (to open the editor as quickly as possible).

https://gitforwindows.org/

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.**


Git 2.24.0.2 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.

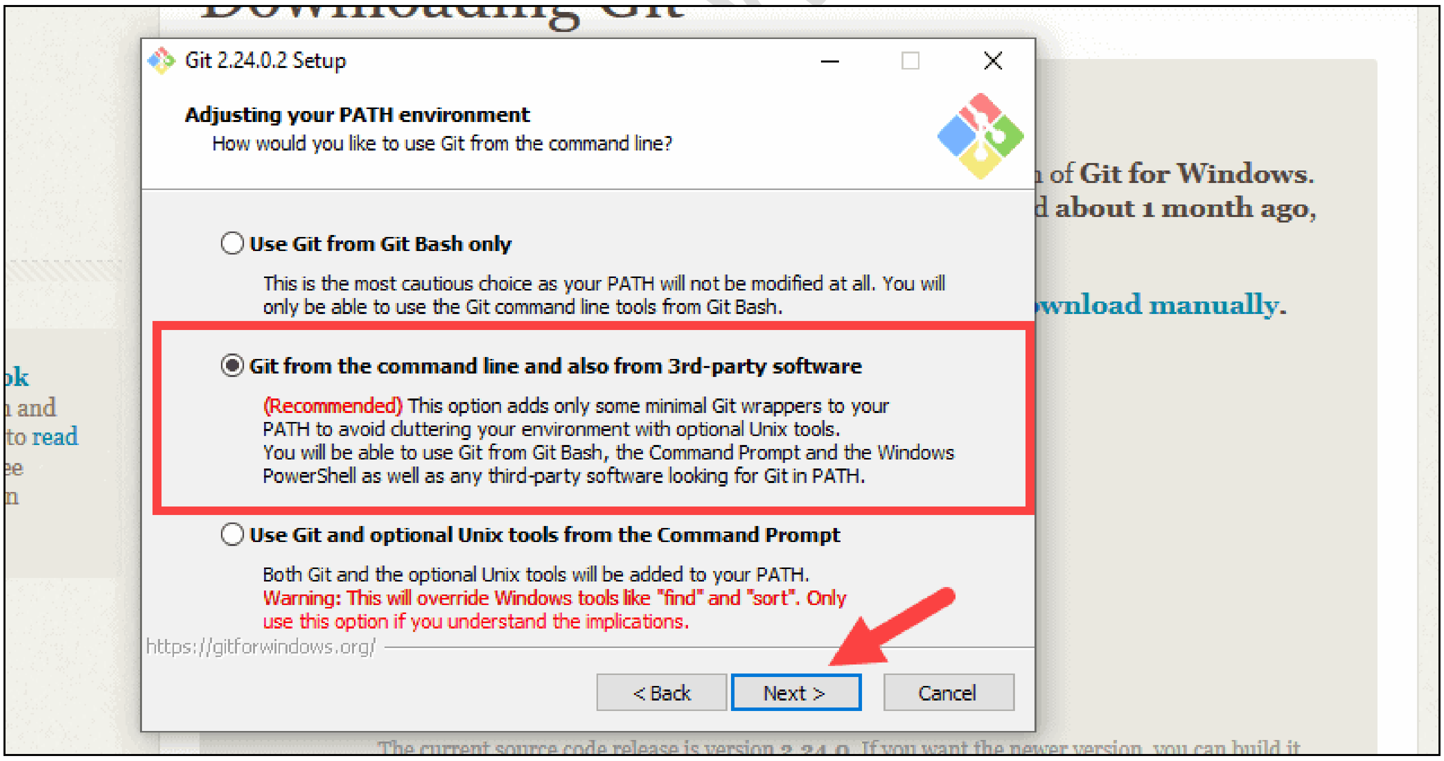
Override the default branch name for new repositories

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:

main

This setting does not affect existing repositories.

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**.


Git 2.24.0.2 Setup

Adjusting your PATH environment

How would you like to use Git from the command line?

Use Git from Git Bash only

This is the most cautious choice as your PATH will not be modified at all. You will only be able to use the Git command line tools from Git Bash.

[✓ Git from the command line and also from 3rd-party software]

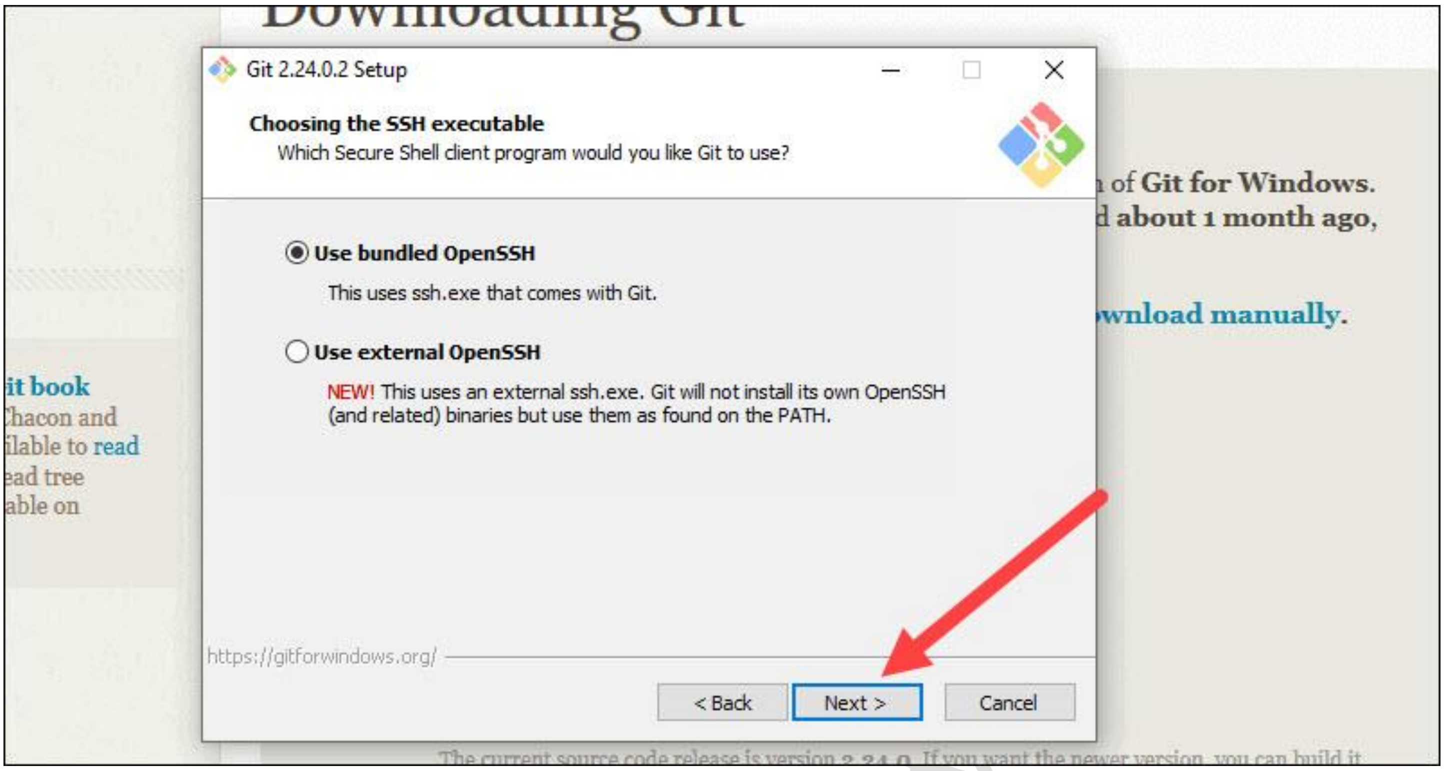
(Recommended) This option adds only some minimal Git wrappers to your PATH to avoid cluttering your environment with optional Unix tools. You will be able to use Git from Git Bash, the Command Prompt and the Windows PowerShell as well as any third-party software looking for Git in PATH.

Use Git and optional Unix tools from the Command Prompt

Both Git and the optional Unix tools will be added to your PATH. Warning: This will override Windows tools like "find" and "sort". Only use this option if you understand the implications.

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.**



Git 2.24.0.2 Setup

Choosing the SSH executable

Which Secure Shell client program would you like Git to use?

[✓ **Use bundled OpenSSH**]

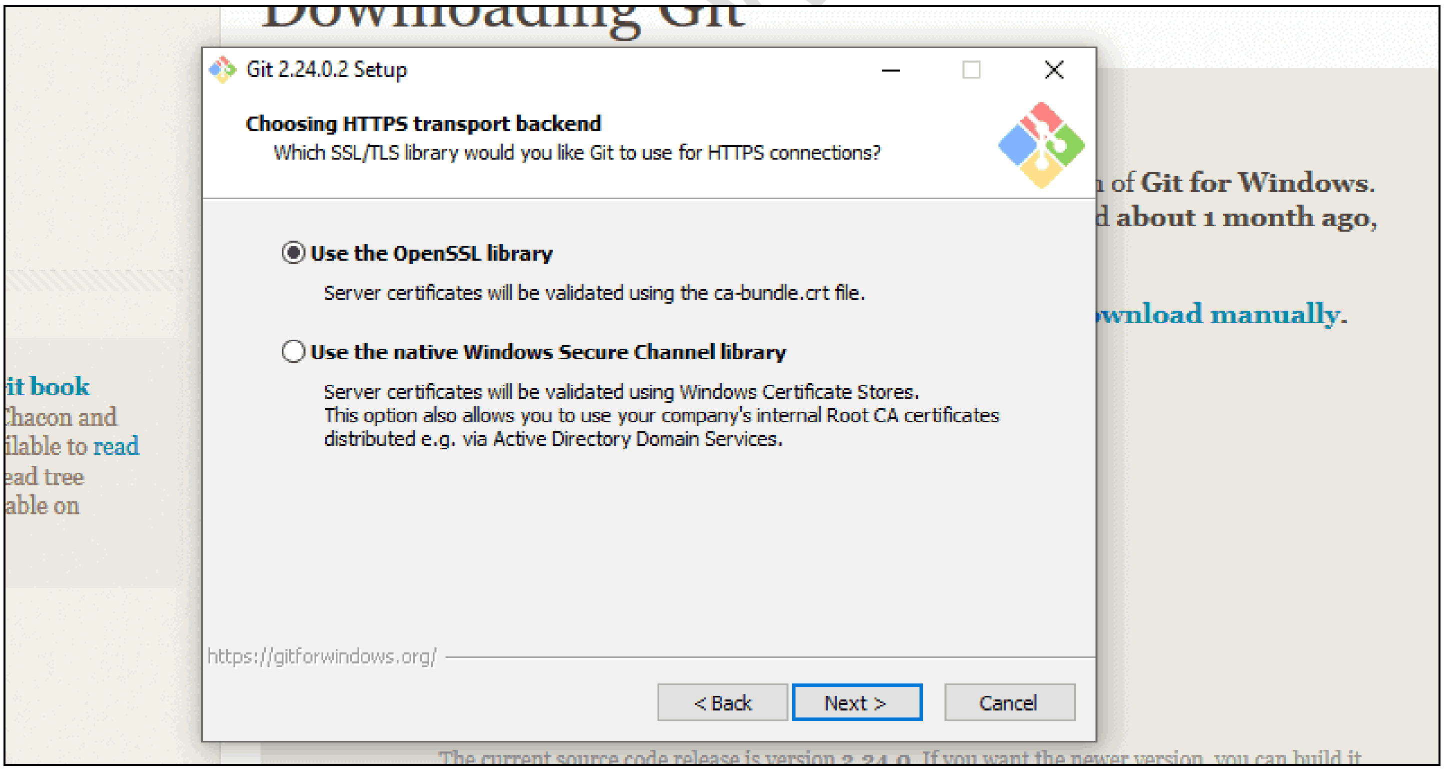
This uses ssh.exe that comes with Git.

Use external OpenSSH

NEW! This uses an external ssh.exe. Git will not install its own OpenSSH (and related) binaries but use them as found on the PATH.

https://gitforwindows.org/

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**.



Git 2.24.0.2 Setup

Choosing HTTPS transport backend

Which SSL/TLS library would you like Git to use for HTTPS connections?

[✓ **Use the OpenSSL library**]

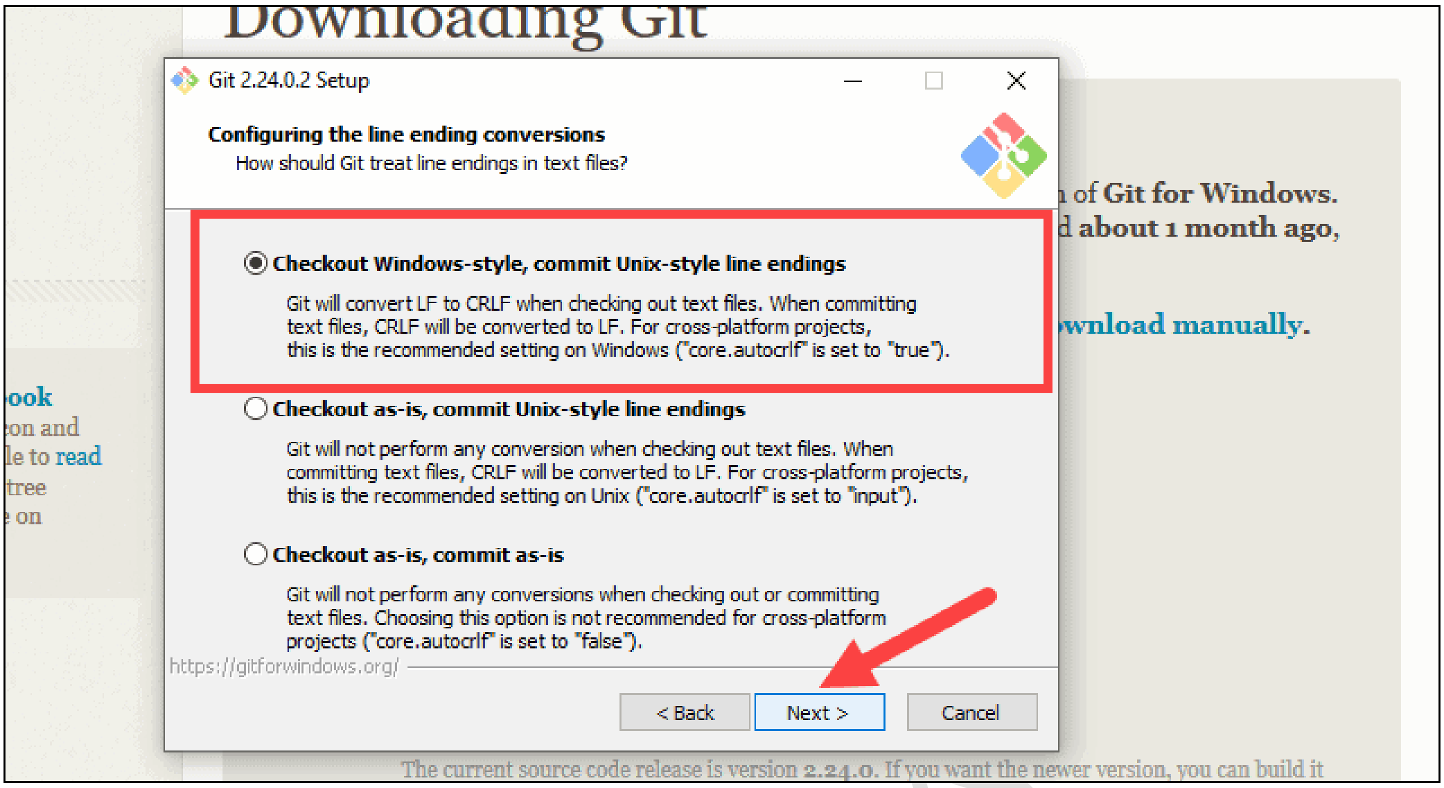
Server certificates will be validated using the ca-bundle.crt file.

Use the native Windows Secure Channel library

Server certificates will be validated using Windows Certificate Stores.
This option also allows you to use your company's internal Root CA certificates distributed e.g. via Active Directory Domain Services.

https://gitforwindows.org/

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**.


Git 2.24.0.2 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").

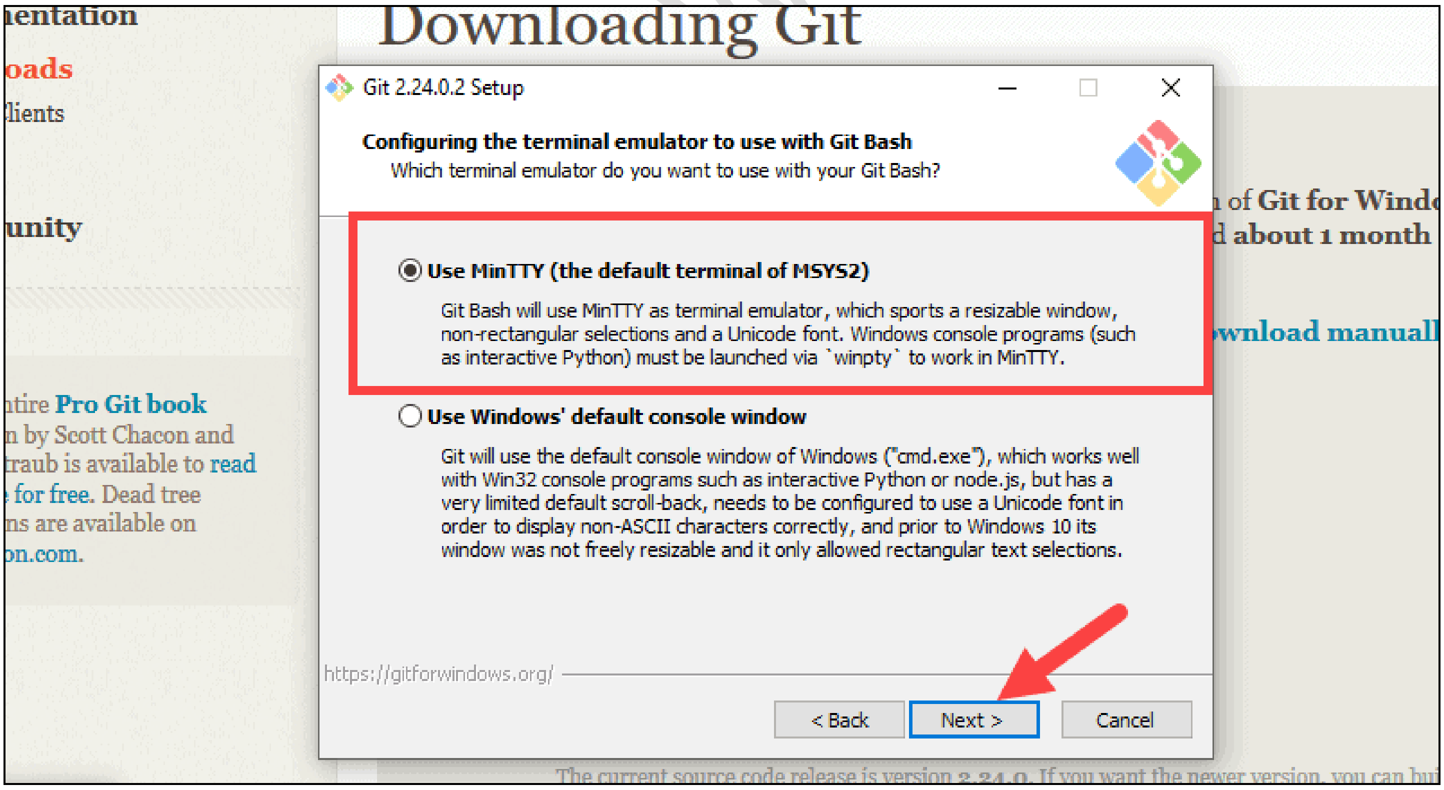
Checkout as-is, commit Unix-style line endings

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").

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


Git 2.24.0.2 Setup

Configuring the terminal emulator to use with Git Bash

Which terminal emulator do you want to use with your Git Bash?

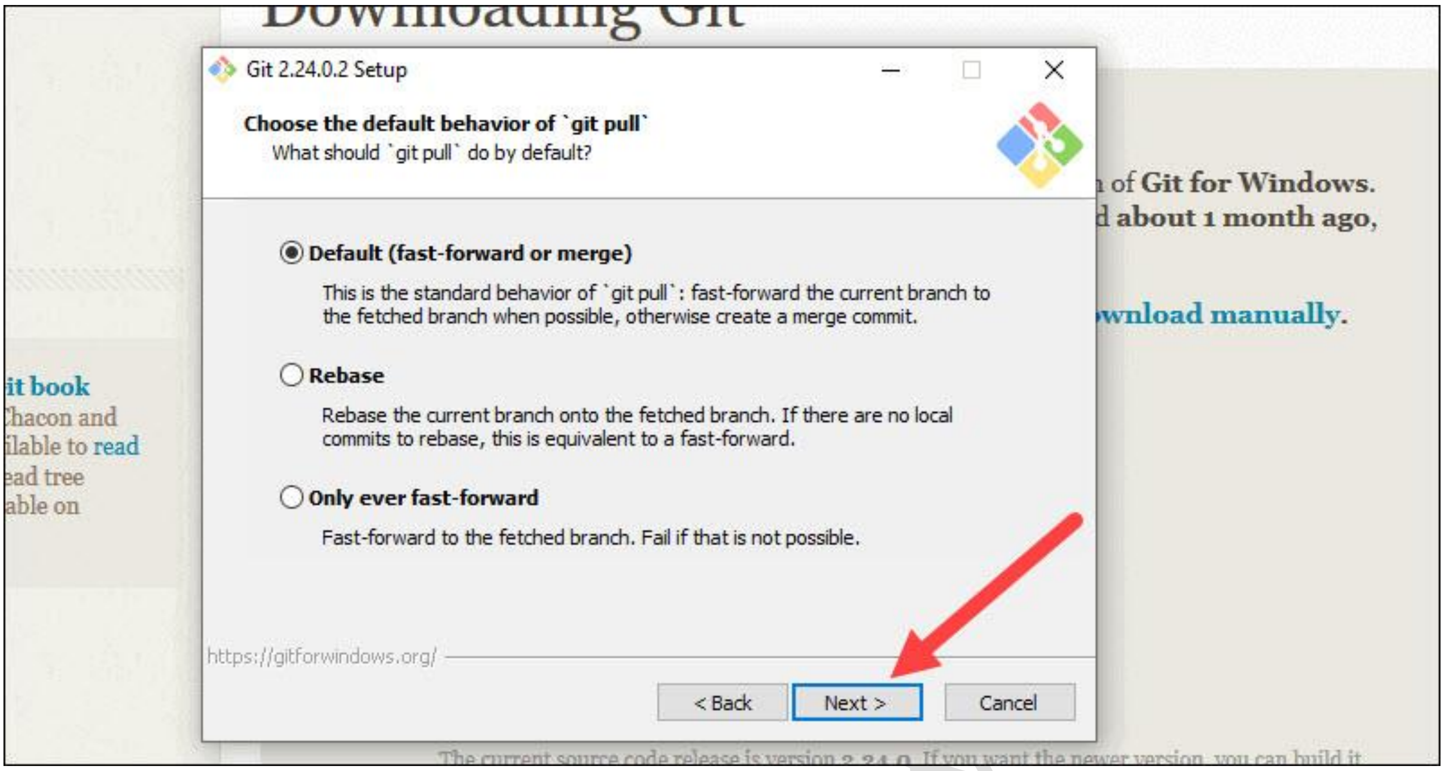
[✓ Use MinTTY (the default terminal of MSYS2)]

Git Bash will use MinTTY as terminal emulator, which sports a resizable window, non-rectangular selections and a Unicode font. Windows console programs (such as interactive Python) must be launched via `winpty` to work in MinTTY.

Use Windows' default console window

Git will use the default console window of Windows ("cmd.exe"), which works well with Win32 console programs such as interactive Python or node.js, but has a very limited default scroll-back, needs to be configured to use a Unicode font in order to display non-ASCII characters correctly, and prior to Windows 10 its window was not freely resizable and it only allowed rectangular text selections.

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



Choose the default behavior of `git pull`

What should `git pull` do by default?

[✓ Default (fast-forward or merge)]

This is the standard behavior of `git pull`: fast-forward the current branch to the fetched branch when possible, otherwise create a merge commit.

Rebase

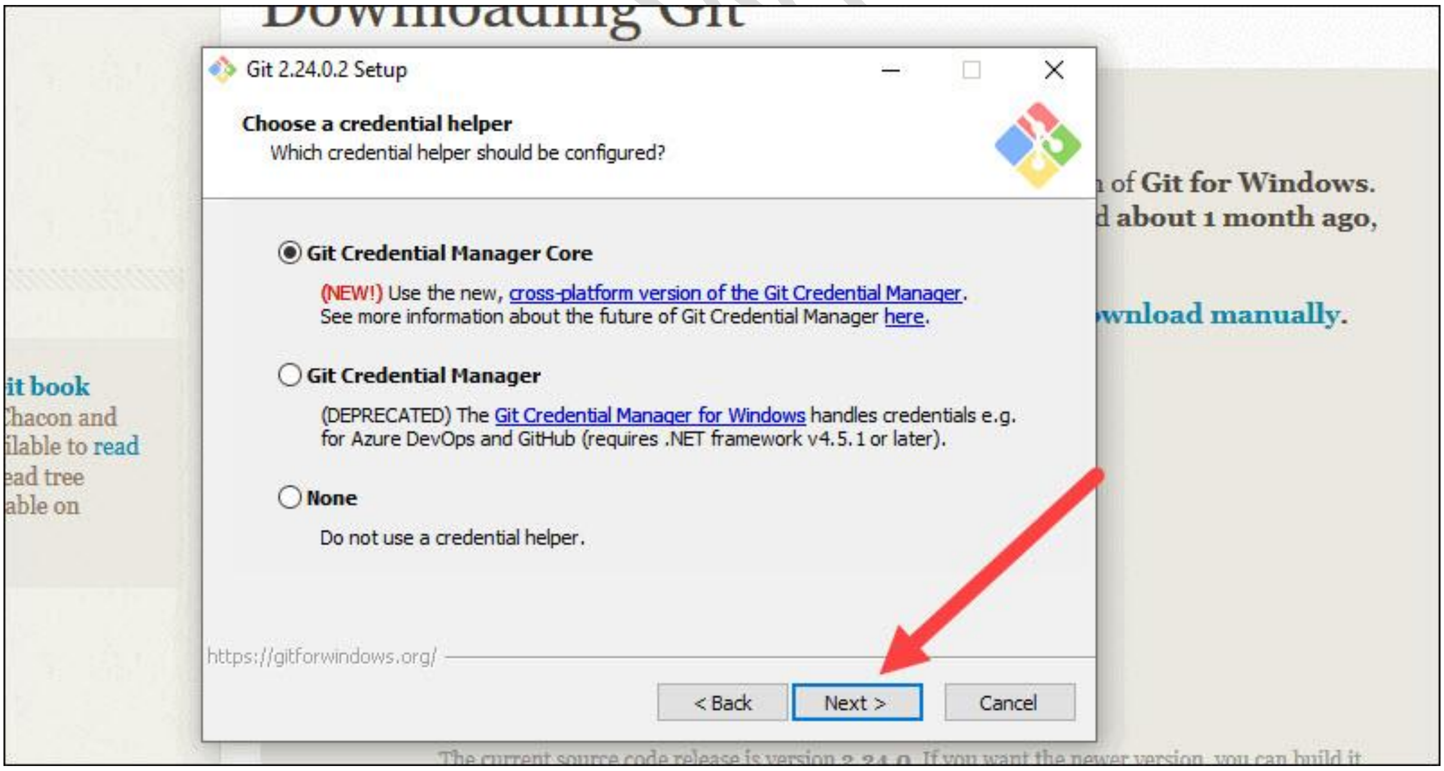
Rebase the current branch onto the fetched branch. If there are no local commits to rebase, this is equivalent to a fast-forward.

Only ever fast-forward

Fast-forward to the fetched branch. Fail if that is not possible.

https://gitforwindows.org/

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**.



Choose a credential helper

Which credential helper should be configured?

[✓ Git Credential Manager Core]

(NEW!) Use the new, cross-platform version of the Git Credential Manager. See more information about the future of Git Credential Manager here.

Git Credential Manager

(DEPRECATED) The Git Credential Manager for Windows handles credentials e.g. for Azure DevOps and GitHub (requires .NET framework v4.5.1 or later).

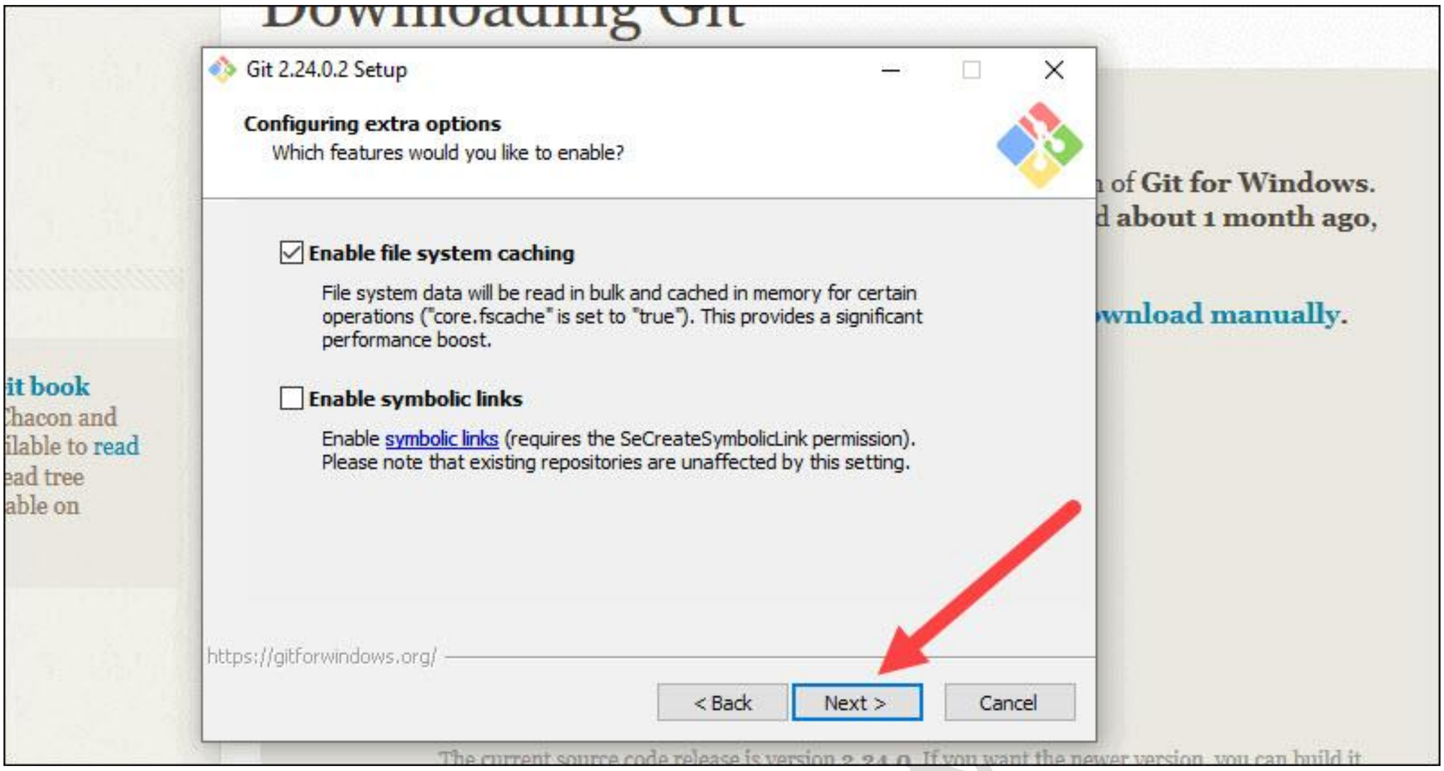
None

Do not use a credential helper.

https://gitforwindows.org/

Additional Customization Options

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**.



Git 2.24.0.2 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.

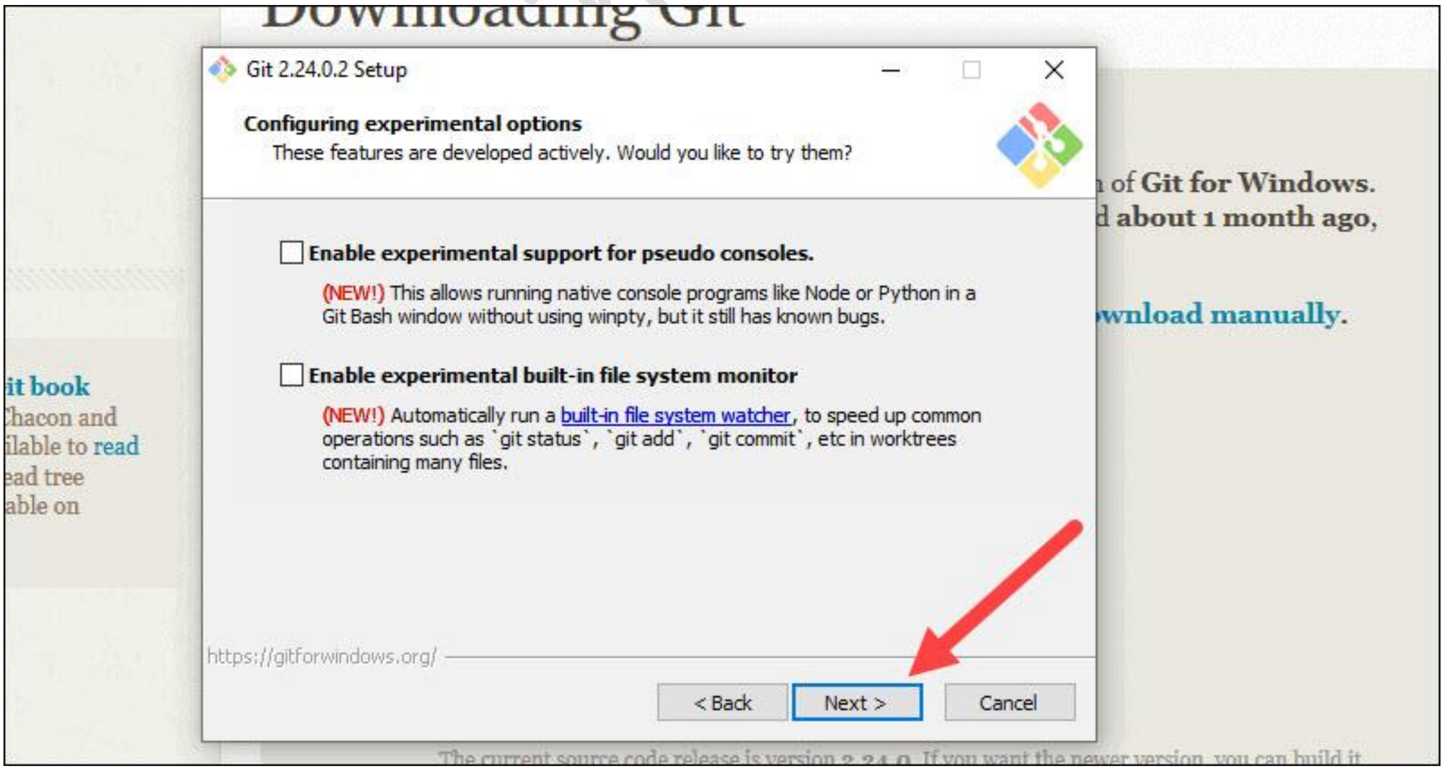
[ ] **Enable symbolic links**

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

https://gitforwindows.org/

< Back | Next > | Cancel

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**.



Git 2.24.0.2 Setup

Configuring experimental options

These features are developed actively. Would you like to try them?

[ ] **Enable experimental support for pseudo consoles.**

(NEW!) This allows running native console programs like Node or Python in a Git Bash window without using winpty, but it still has known bugs.

[ ] **Enable experimental built-in file system monitor**

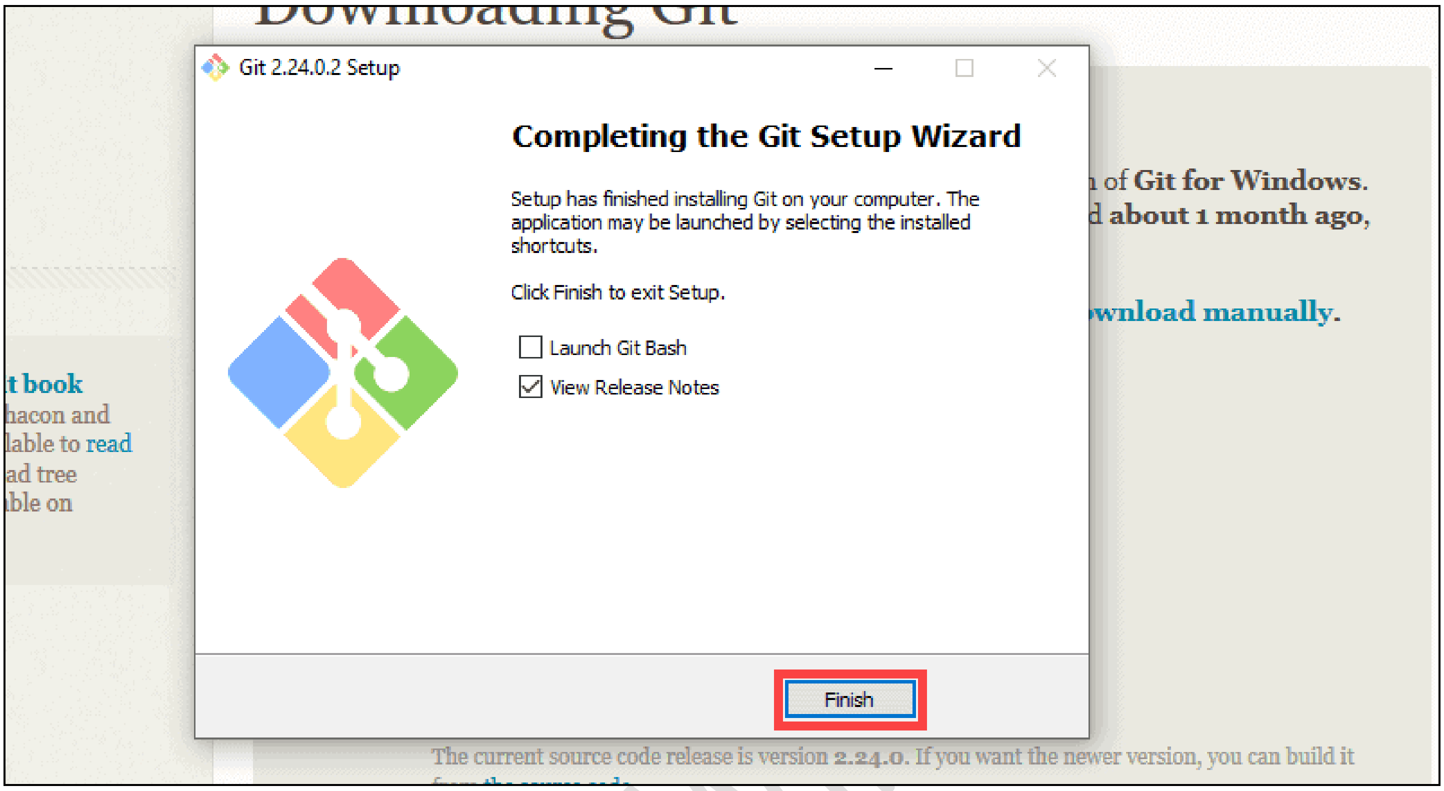
(NEW!) Automatically run a built-in file system watcher, to speed up common operations such as `git status`, `git add`, `git commit`, etc in worktrees containing many files.

https://gitforwindows.org/

< Back | Next > | Cancel

Complete Git Installation Process

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


Git 2.24.0.2 Setup

Completing the Git Setup Wizard

Setup has finished installing Git on your computer. The application may be launched by selecting the installed shortcuts.

Click Finish to exit Setup.

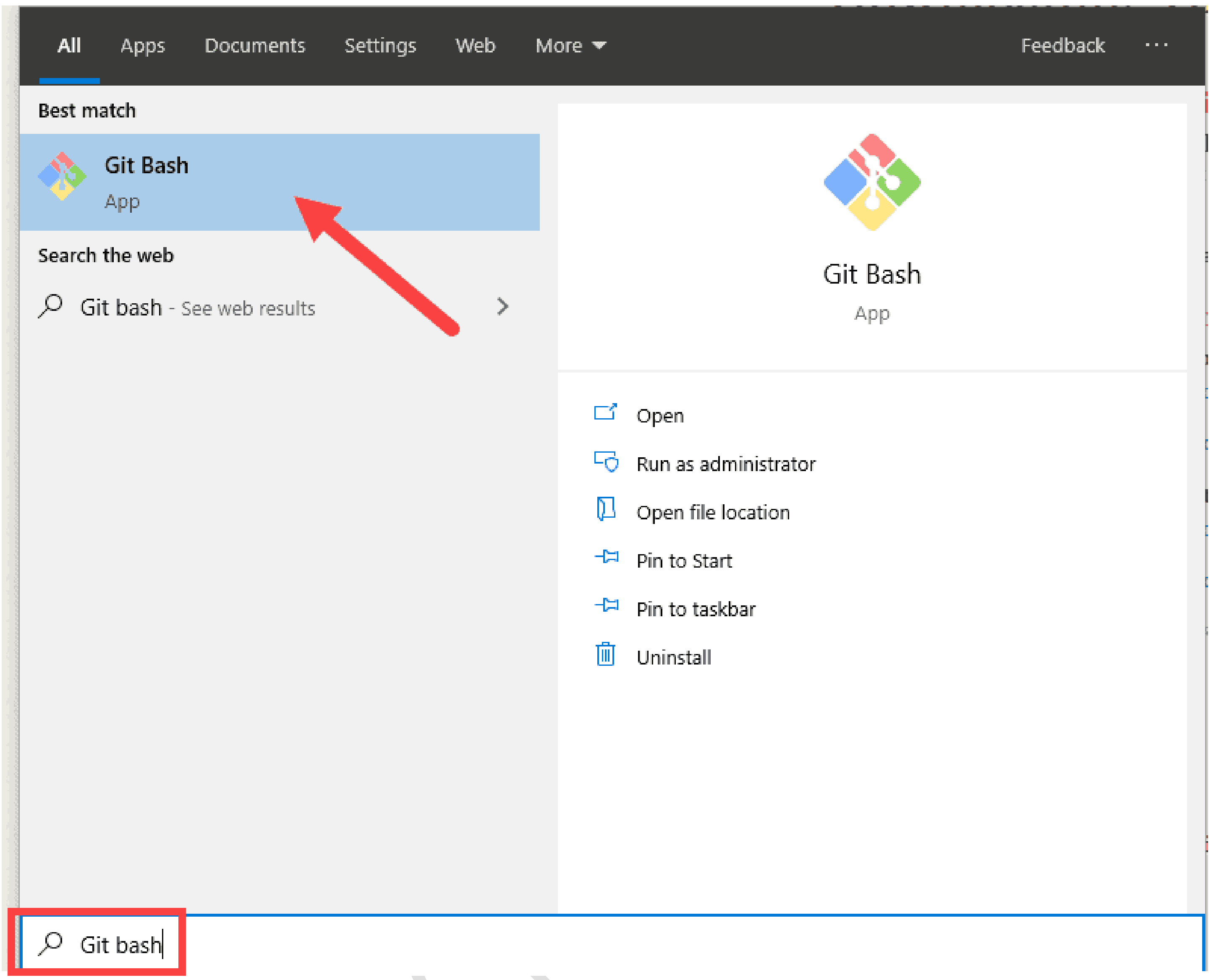
☐ Launch Git Bash
☑ View Release Notes

How to Launch Git in Windows

Git has two modes of use – a **bash scripting shell** (or command line) and a **graphical user interface** (GUI).

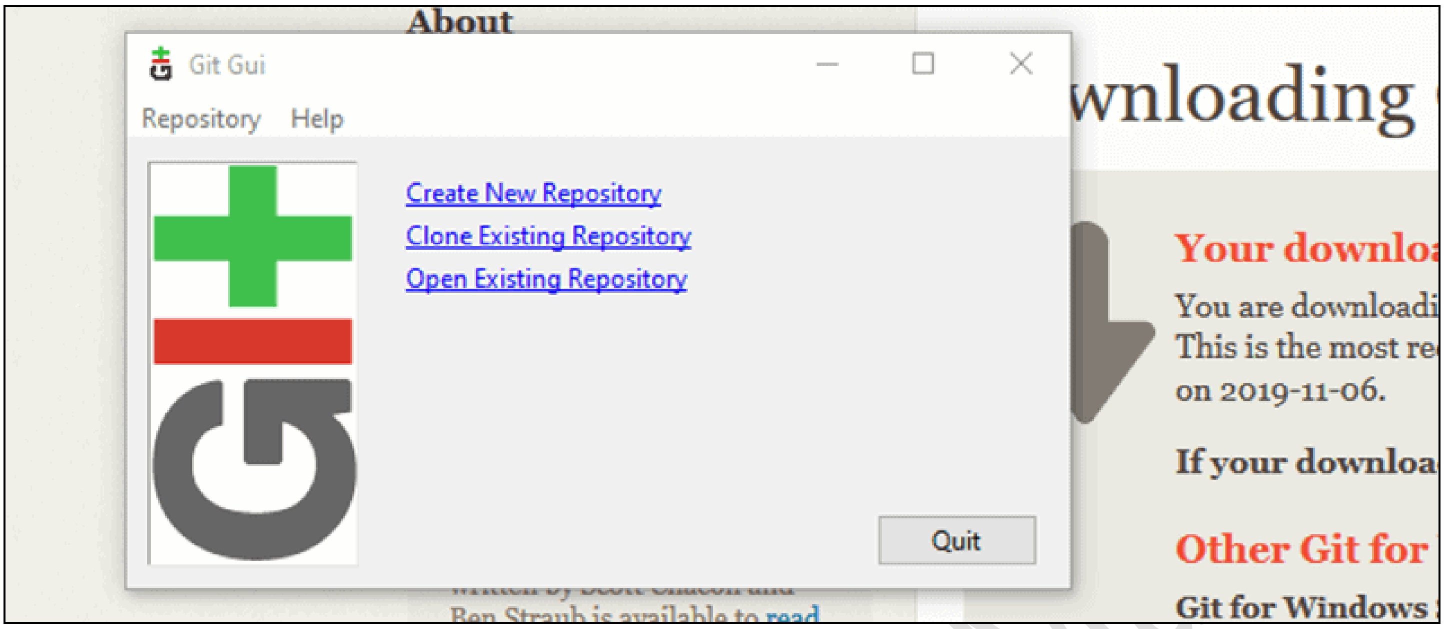
Launch Git Bash Shell

To launch **Git Bash** open the **Windows Start** menu, type ***git bash*** and press **Enter** (or click the application icon).



Launch Git GUI

To launch **Git GUI** open the **Windows Start** menu, type ***git gui*** and press **Enter** (or click the application icon).



Connecting to a Remote Repository

You need a GitHub username and password for this next step.

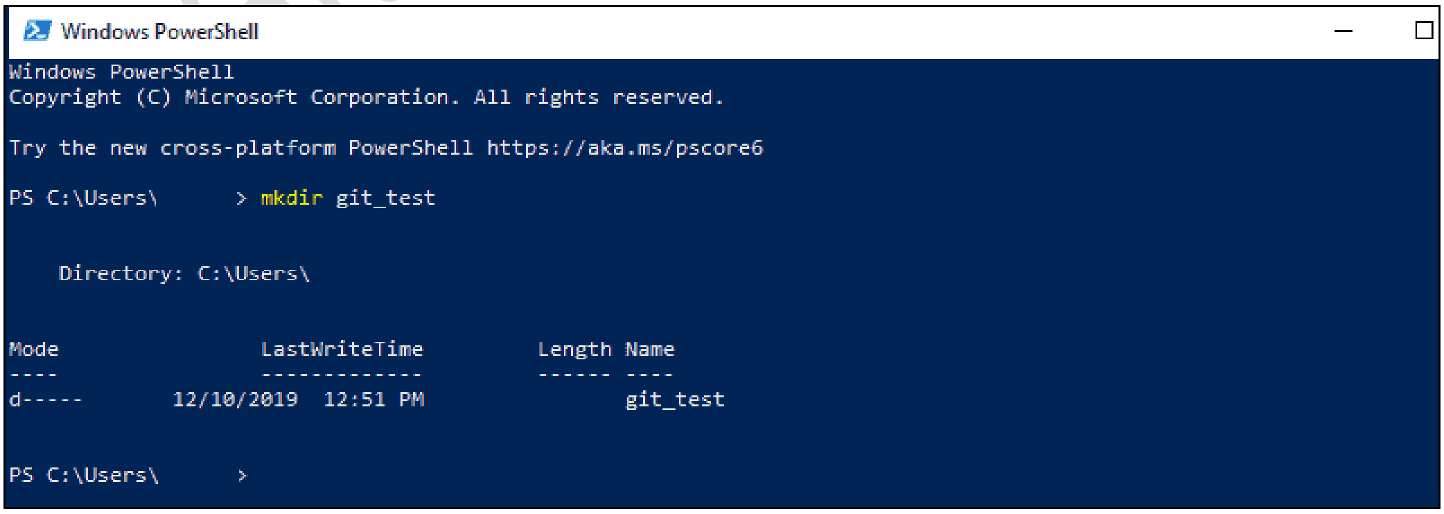
Create a Test Directory

Open a Windows PowerShell interface by pressing **Windows Key + x**, and then **i** once the menu appears.

Create a new test directory (folder) by entering the following:

```
mkdir git_test
```

An example of the PowerShell output.



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\      > mkdir git_test


    Directory: C:\Users\


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        12/10/2019  12:51 PM                git_test


PS C:\Users\      >
```

Change your location to the newly created directory:

```
cd git_test
```

Note: If you already have a GitHub repository, use the name of that project instead of **git_test**.

Configure GitHub Credentials

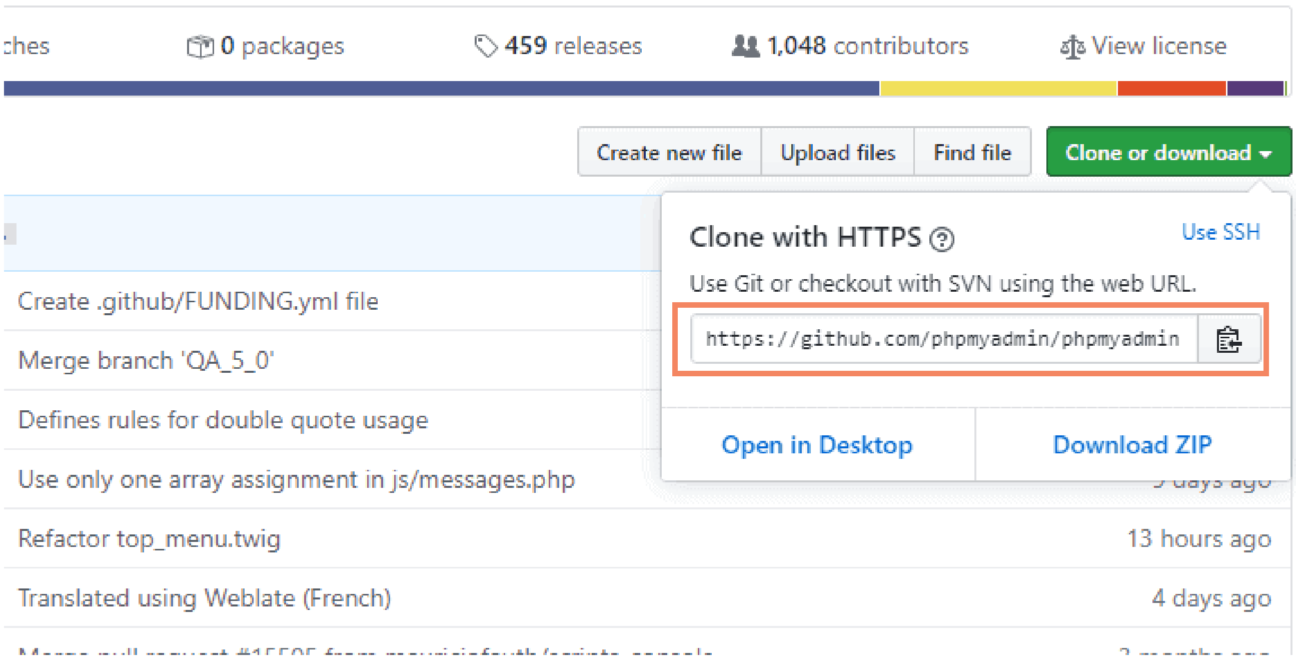
Configure your local Git installation to use your GitHub credentials by entering the following:

```
git config --global user.name "github_username"
git config --global user.email "email_address"
```

Note: Replace **github_username** and **email_address** with your GitHub credentials.

Clone a GitHub Repository

Go to your repository on GitHub. In the top right above the list of files, open the **Clone or download** drop-down menu. Copy the **URL for cloning over HTTPS**.



0 packages
459 releases
1,048 contributors
View license
Create new file
Upload files
Find file
Clone or download
Clone with HTTPS
Use SSH
Use Git or checkout with SVN using the web URL.
https://github.com/phpmyadmin/phpmyadmin
Open in Desktop
Download ZIP
Create .github/FUNDING.yml file
Merge branch 'QA_5_0'
Defines rules for double quote usage
Use only one array assignment in js/messages.php
Refactor top_menu.twig
13 hours ago
Translated using Weblate (French)
4 days ago

Switch to your PowerShell window, and enter the following:

```
git clone repository_url
```

Important: In the example above, the command will clone the repository over HTTPS. Another option is **cloning with SSH URLs**. For that option to work, you must generate an SSH key pair on your Windows workstation and assign the public key to your GitHub account.

List Remote Repositories

Your working directory should now have a copy of the repository from GitHub. It should contain a directory with the name of the project. Change to the directory:

```
cd git_project
```

Note: Replace **git_project** with the actual name of the repository you downloaded. If it's not working, you can list the contents of the current directory with the **ls command**. This is helpful if you don't know the exact name or need to check your spelling.

Once you're in the sub-directory, list the remote repositories:

```
git remote -v
```

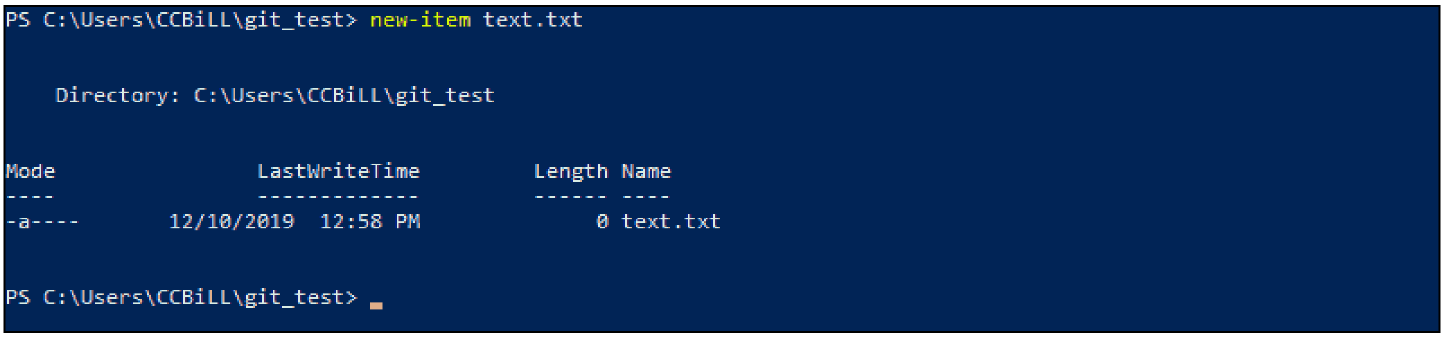
Pushing Local Files to the Remote Repository

Once you've done some work on the project, you may want to submit those changes to the remote project on GitHub.

1. For example, create a new text file by entering the following into your PowerShell window:

```
new-item text.txt
```

2. Confirmation that the new file is created.



```
PS C:\Users\CCBiLL\git_test> new-item text.txt


    Directory: C:\Users\CCBiLL\git_test


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----        12/10/2019  12:58 PM              0 text.txt


PS C:\Users\CCBiLL\git_test> _
```

3. Now check the status of your new Git branch and untracked files:

```
git status
```

4. Add your new file to the local project:

```
git add text.txt
```

5. Run **git status** again to make sure the text.txt file has been added. Next, commit the changes to the local project:

```
git commit -m "Sample 1"
```

6. Finally, push the changes to the remote GitHub repository:

```
git push
```

You may need to enter your username and password for GitHub.

Note: You can remove a remote repository if the need for it no longer exists. To learn how, visit our guide How to Remove a Git Remote.

Git set up at Mac:

Pre-requisites

- A MacOS
- Access to command line/terminal window

How to Install Git on Mac

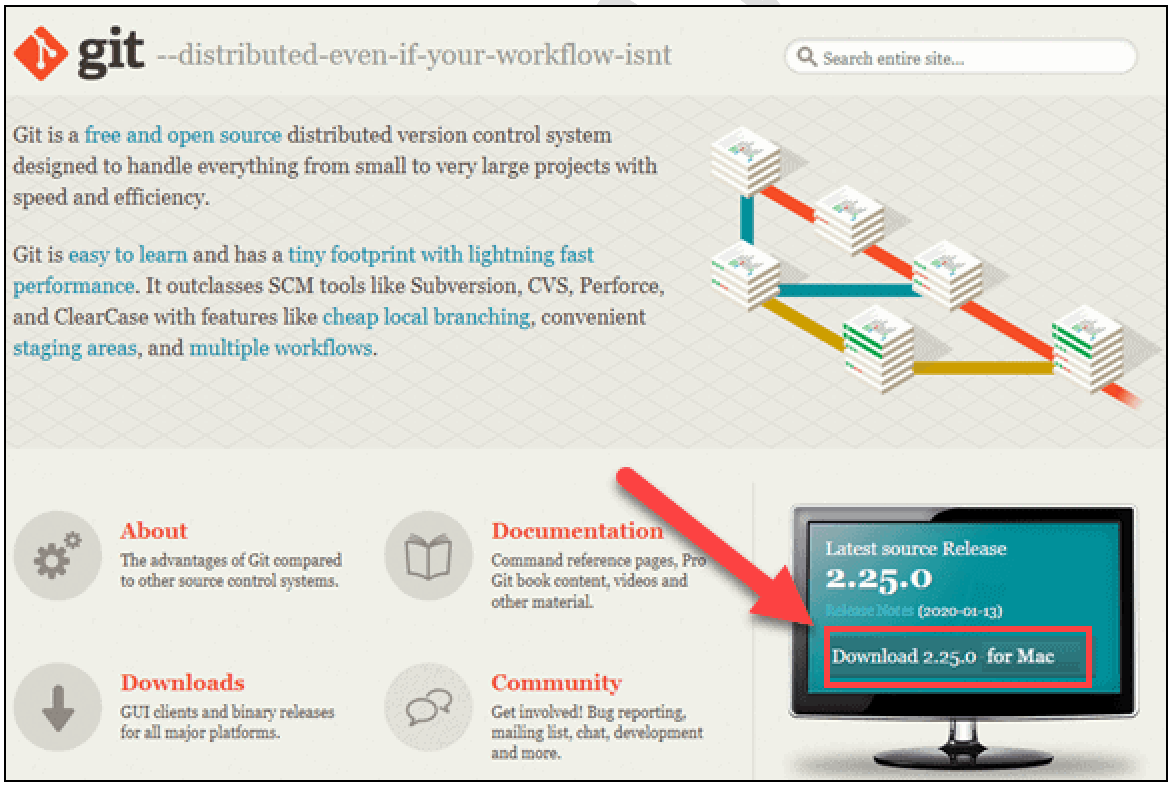
There are many different ways to set up Git on Mac. If you prefer using a GUI, Git offers a simple installation using the installer for Mac. On the other hand, you can install Git using the terminal with a couple of simple commands.

Option 1: Install Git on Mac with Installer

The easiest way to set up Git is to use the Git installer for Mac.

1. Open a browser and navigate to Git's official website.

2. You will see a display showing the version number of the latest source release and a download button, as in the image below.



git --distributed-even-if-your-workflow-isnt

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.

Git is easy to learn and has a tiny footprint with lightning fast performance. It outclasses SCM tools like Subversion, CVS, Perforce, and ClearCase with features like cheap local branching, convenient staging areas, and multiple workflows.

About
The advantages of Git compared to other source control systems.

Documentation
Command reference pages, Pro Git book content, videos and other material.

Downloads
GUI clients and binary releases for all major platforms.

Community
Get involved! Bug reporting, mailing list, chat, development and more.

Latest source Release
2.25.0
Release Notes (2020-01-13)
Download 2.25.0 for Mac

3. Click **Download**, and it automatically downloads the software package on your system.

4. Find the package and double-click to open the **Git installer**.

5. Follow the **installation wizard** and configure Git to suit your development needs. If you are new to version control systems, the best option would be to leave the default settings.

6. Click **Install** and type in your password if necessary.

7. Confirm once again by clicking **Install Software**.

With this, you have finished setting up Git on your Mac. Move on to the next step of configuring Git.

Option 2: Install Git on Mac using the Terminal

There are multiple ways to install Git on Mac via terminal, depending on the development environment or package manager you have on your system.

This guide includes three different options.

Install Git Using Xcode

If you prefer the terminal, using **Xcode** is the fastest and easiest way to start working with Git. Its command-line tools include Git in the package.

Users who don't have Xcode can install it with a single command:

```
xcode-select --install
```

With Xcode running on your Mac, you can check whether Git is also available by prompting for the **Git version**:

```
git --version
```

The output should display the latest Git release, as in the example below.

```
git version 2.25.0 (Apple Git-66)
```

If you do not have Git, it automatically asks you whether you want to install it. Confirm the installation, and Xcode sets up Git.

Install Git Using Homebrew

Another way to install Git is with **Homebrew**, the package management system for Mac.

Run the following **brew** command in the terminal:

```
brew install git
```

Then, **check the Git version** to verify the installation:

```
git --version
```

Install Git Using MacPorts

If you are using **MacPorts** to manage your packages on the system, you can use the **port** command to set up Git.

Start by **updating MacPorts** with the command:

```
sudo port selfupdate
```

Search for and install the newest **Git ports and variants** by running the following two commands:

```
port search git
port variants git
```

Then, **install Git** with:

```
sudo port install git
```

Note: When setting up Git with MacPorts, you can **install additional tools** you may find useful in the future. Add the bash-completion, svn, and the docs to the command for installing: **sudo port install git +svn +doc +bash_completion +gitweb**

Get Started with Git on Mac

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"
```

Track and Commit Changes

To demonstrate how to work with files on local Git repositories, we are going to create a demo folder and file to work with.

1. First, open the terminal and create a new folder named ***NewFolder***.

```
mkdir /Users/[username]/Desktop/Tools/Git/NewFolder
```

2. Then, move into that directory. The path may differ according to the location where you created the new folder.

```
cd /Users/[username]/Desktop/Tools/Git/NewFolder/
```

3. As we want to keep track of changes inside this folder, we need to create a local Git repository for it. Running the **git init** command initializes an empty git repository in this particular location. Therefore, run the command:

```
git init
```

With this, you have added a hidden folder inside the directory by the name ***.git***.

Note: To see the hidden **.git** folder, you need to run the command: **defaults write com.apple.finder AppleShowAllFiles YES**. If you want to hide the folder again, modify the last part of the command by changing the **YES** to **NO**).

4. While in the directory **NewFolder**, type the following command:

```
git status
```

This shows the state of the working directory and displays if any changes made inside the directory.

Since the folder we created doesn't have any files in it, the output responds with: **nothing to commit**.

5. Add some files inside **NewFolder** and see how the **git status** changes:

```
touch file1.txt
```

6. Check the status again:

```
git status
```

The output tells you there are **untracked files** inside the directory and lists **file1.txt**. Git is tracking the folder in which the file was added, and notifies you that the changes are not being tracked.

7. Prompt Git to track the new file by running:

```
git add test1.txt
```

If you recheck the **git status** now, you would see that the file is now being tracked (as it changed from red to green). However, you still need to **commit** this change.

8. Commit all changes and add a message that describes the commit:

```
git commit -m "added test1.txt"
```

Now, the output tells you the working tree is clean, and there is nothing to commit.