**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)

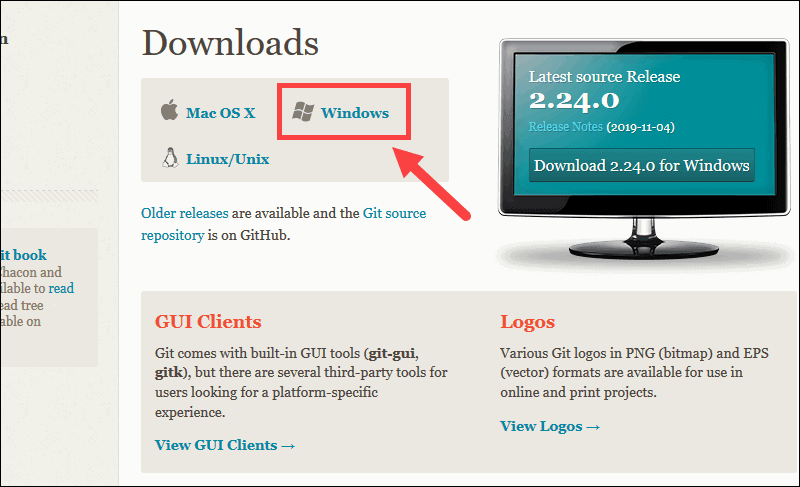
**Steps For Installing Git for Windows**

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.

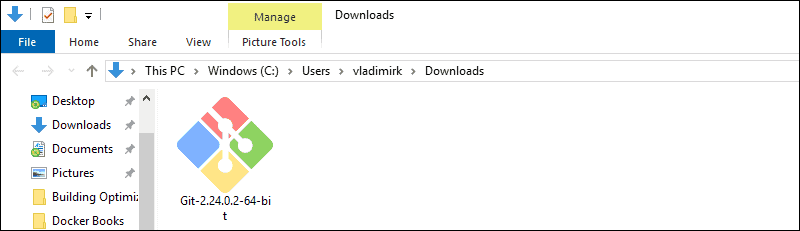
**Download Git for Windows**

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.

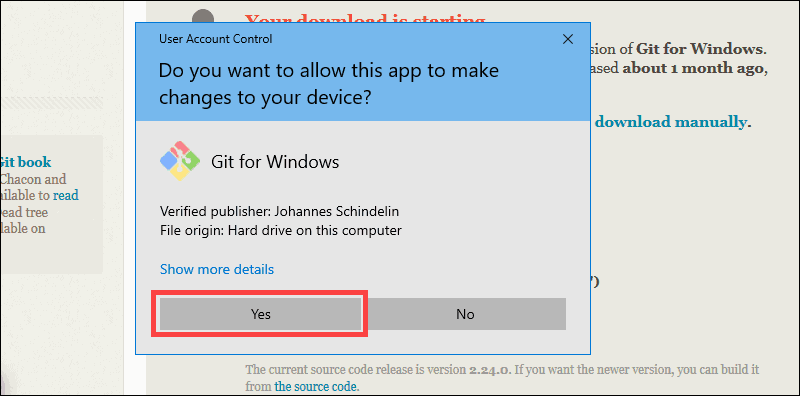


**Extract and Launch Git Installer**

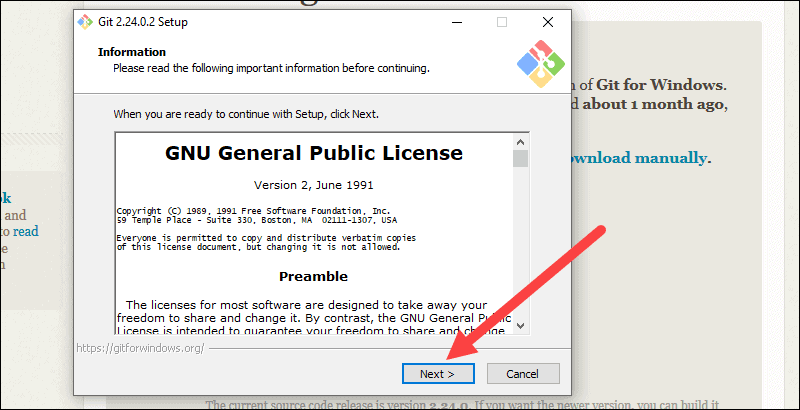
3. Browse to the download location (or use the download shortcut in your browser). Double-click the file to extract and launch the installer.



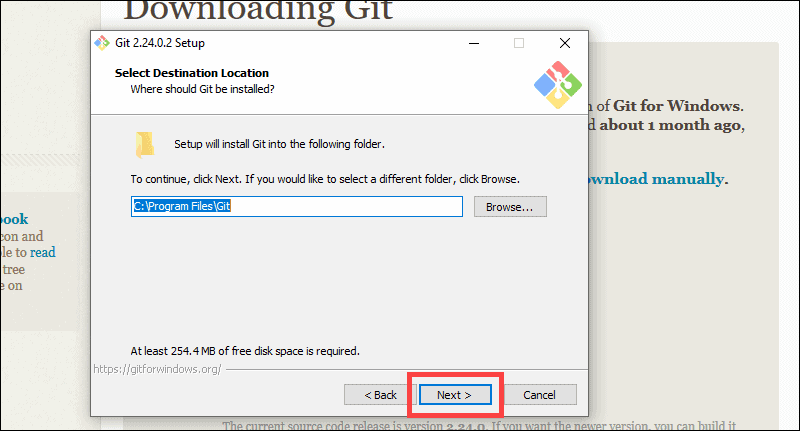
4. Allow the app to make changes to your device by clicking **Yes** on the User Account Control dialog that opens.



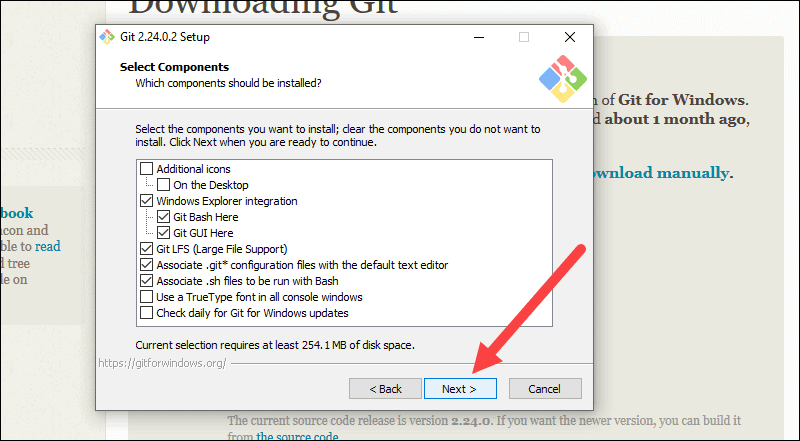
5. Review the GNU General Public License, and when you’re ready to install, click **Next**.



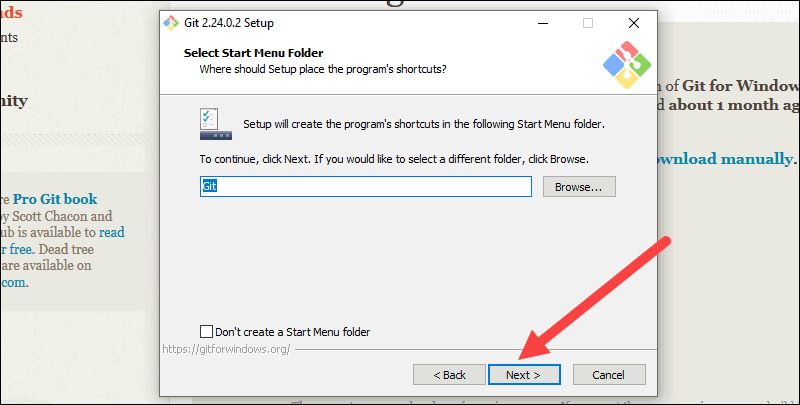
6. The installer will ask you for an installation location. Leave the default, unless you have reason to change it, and click **Next**.



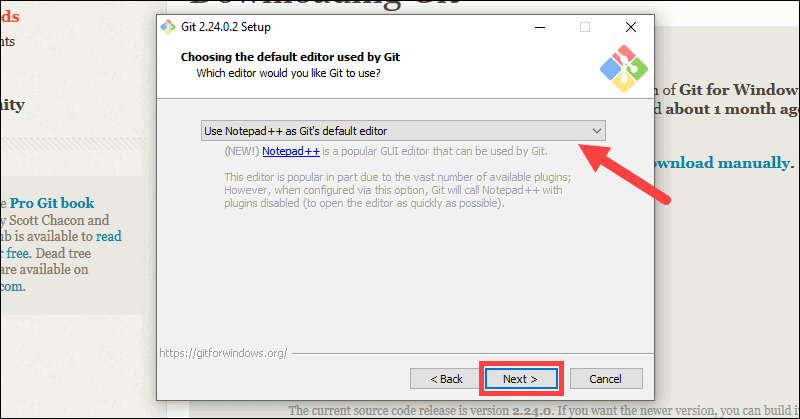
7. A component selection screen will appear. Leave the defaults unless you have a specific need to change them and click **Next**.



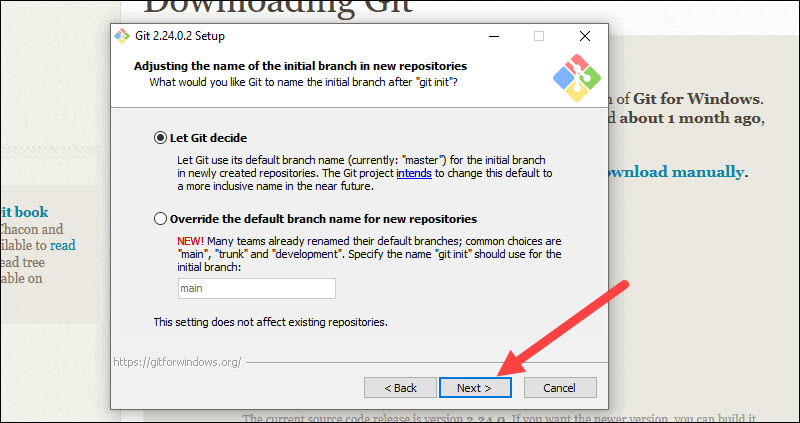
8. The installer will offer to create a start menu folder. Simply click **Next**.



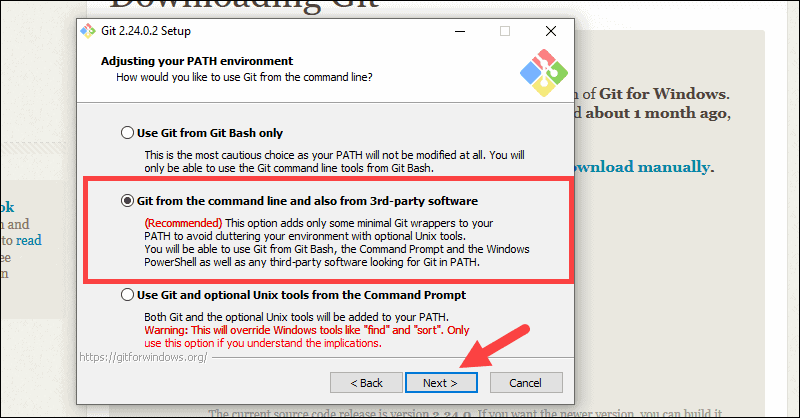
9. Select a text editor you’d like to use with Git. Use the drop-down menu to select Notepad++ (or whichever text editor you prefer) and click **Next**.



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.**

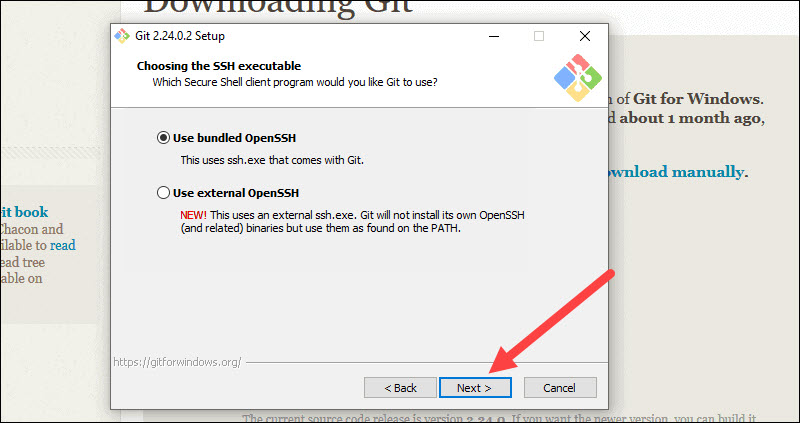


11. This installation step allows you to change the **PATH environment**. The **PATH**is the default set of directories included when you run a command from the command line. Leave this on the middle (recommended) selection and click **Next**.

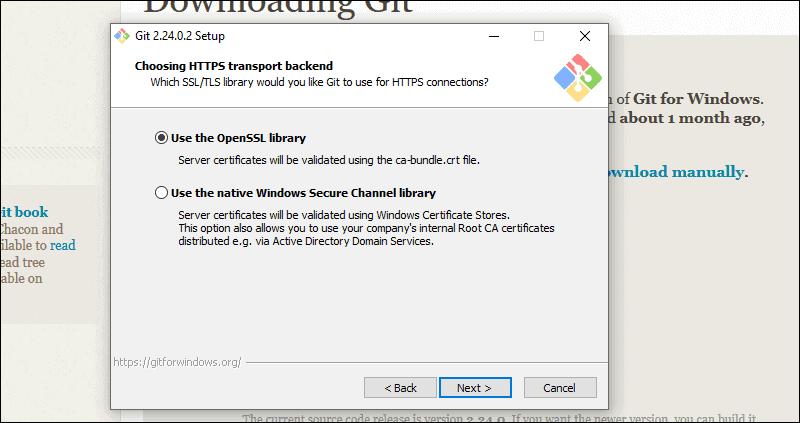


**Server Certificates, Line Endings and Terminal Emulators**

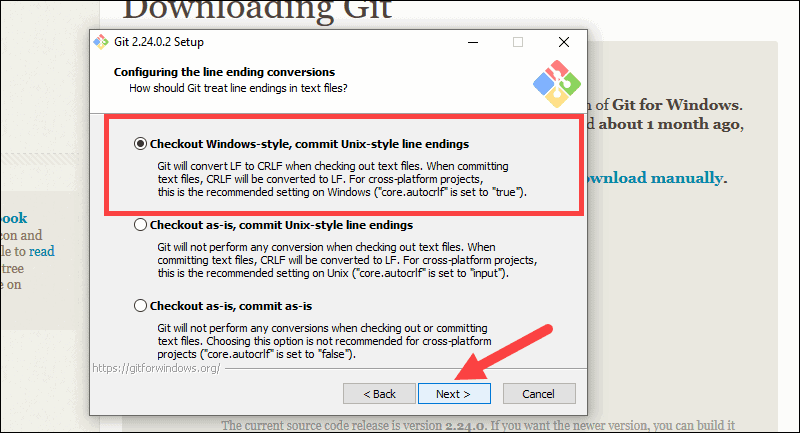
12. The installer now asks which SSH client you want Git to use. Git already comes with its own SSH client, so if you don't need a specific one, leave the default option and click **Next.**



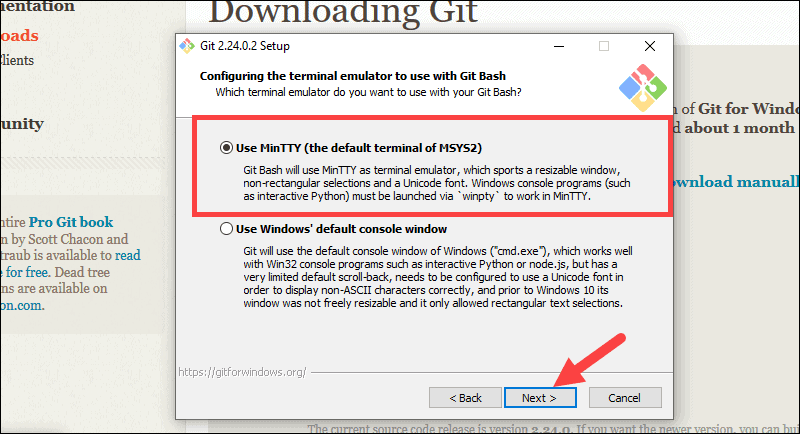
13. The next option relates to server certificates. Most users should use the default. If you’re working in an Active Directory environment, you may need to switch to Windows Store certificates. Click **Next**.



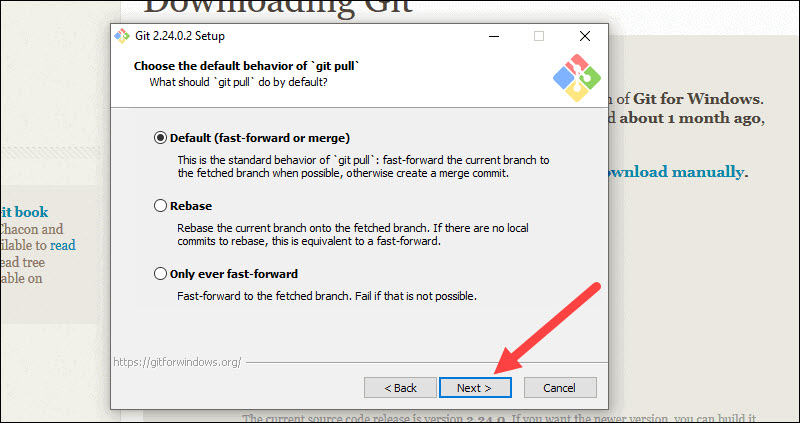
14. The next selection converts line endings. It is recommended that you leave the default selection. This relates to the way data is formatted and changing this option may cause problems. Click **Next**.



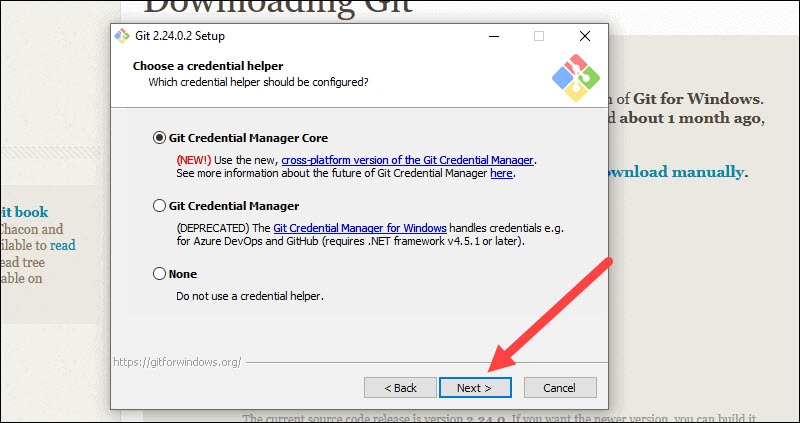
15. Choose the terminal emulator you want to use. The default MinTTY is recommended, for its features. Click **Next**.



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

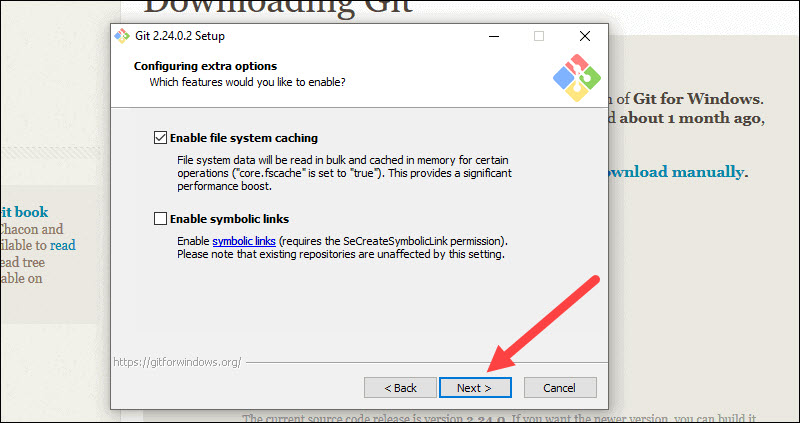


17. Next you should choose which credential helper to use. Git uses credential helpers to fetch or save credentials. Leave the default option as it is the most stable one, and click **Next**.

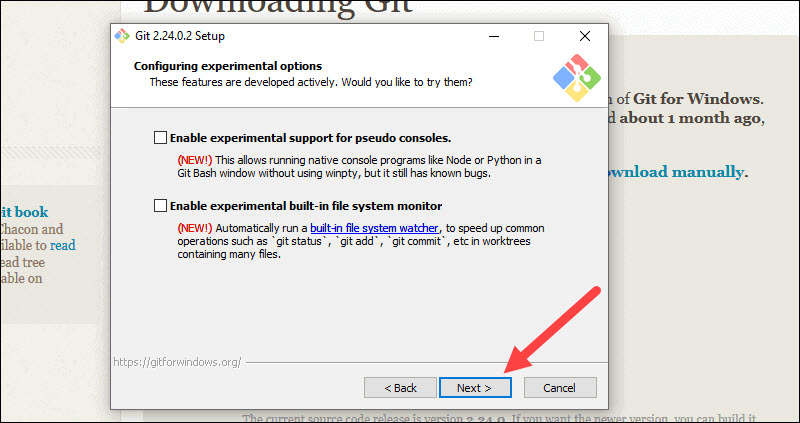


**Additional Customization Options**

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

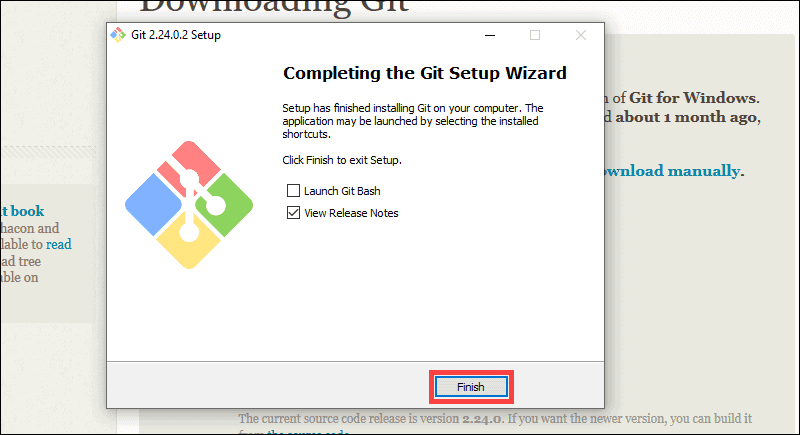


19. Depending on the version of Git you’re installing, it may offer to install experimental features. At the time this article was written, the options to include support for pseudo controls and a built-in file system monitor were offered. Unless you are feeling adventurous, leave them unchecked and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



**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**.

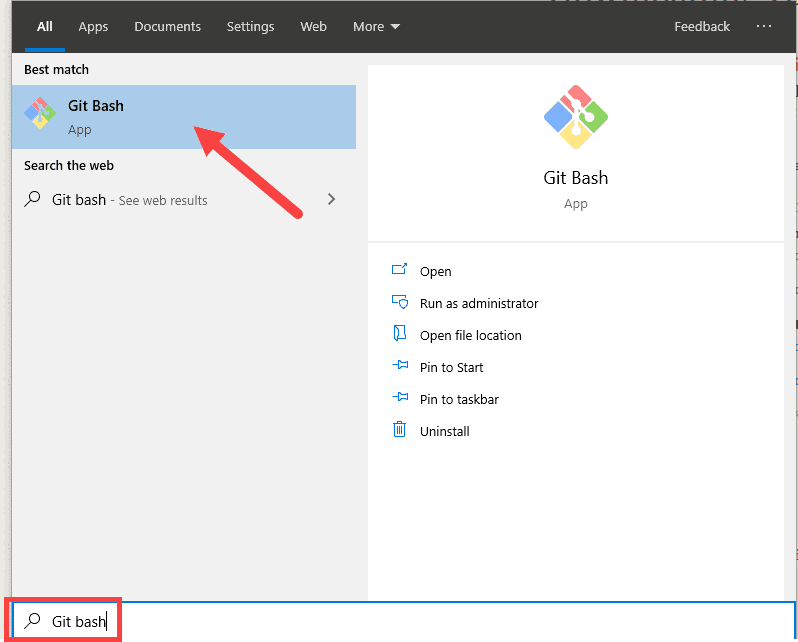


**How to Launch Git in Windows**

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

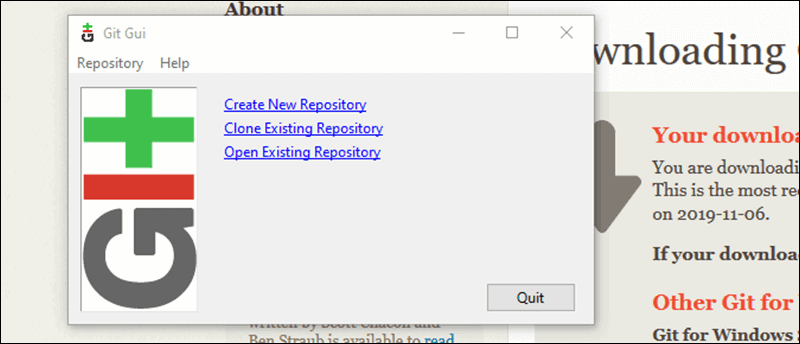
**Launch Git Bash Shell**

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.

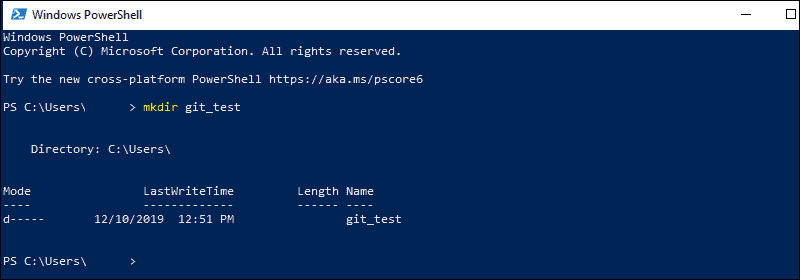
**Create a Test Directory**

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.



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:

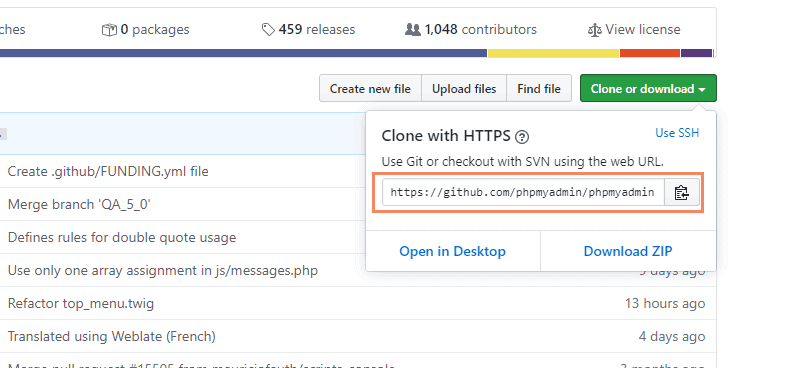
git config --global user.name "github\_username"

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**.



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

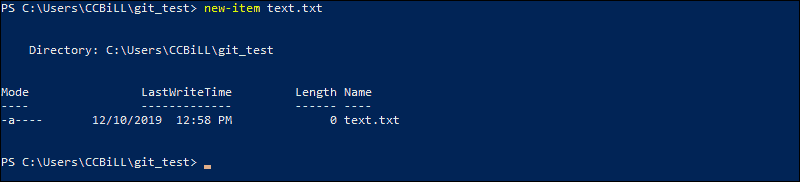
**Pushing Local Files to the Remote Repository**

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.



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

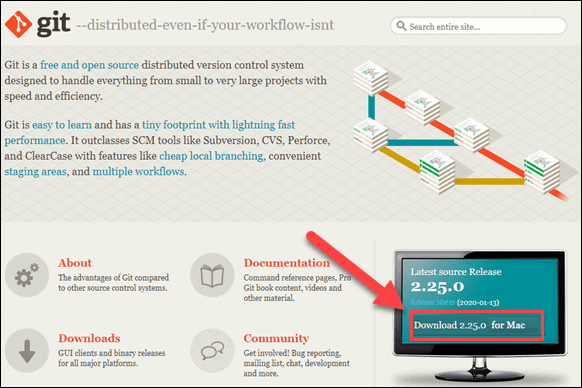
There are many different ways to set up Git on Mac. If you prefer using a GUI, Git offers a simple installation using the installer for Mac. On the other hand, you can install Git using the terminal with a couple of simple commands.

### 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](https://git-scm.com/).

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



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.