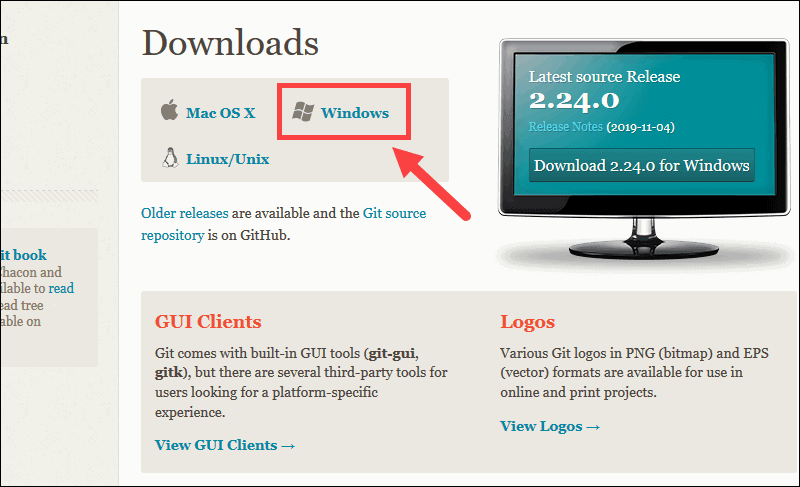
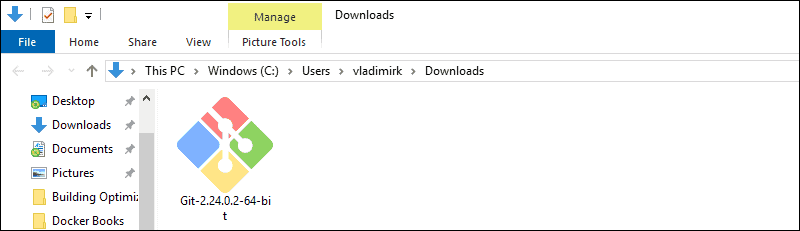
**Download Git for Windows**

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.

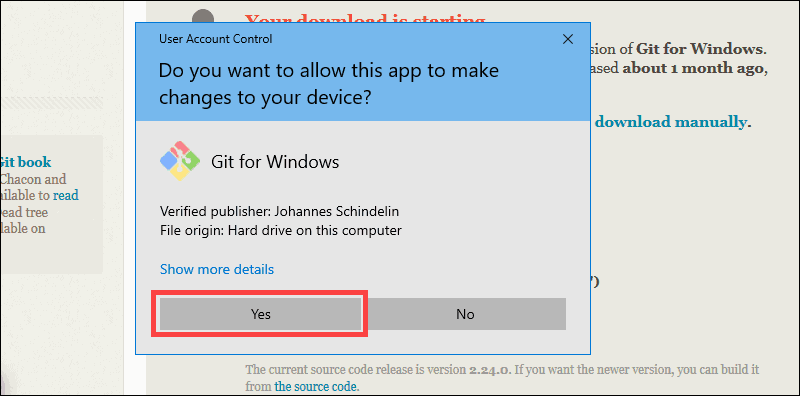


**Extract and Launch Git Installer**

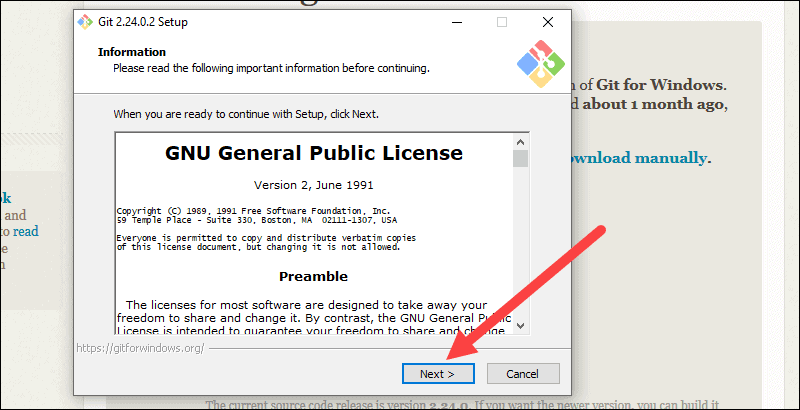
3. Browse to the download location (or use the download shortcut in your browser). Double-click the file to extract and launch the installer.



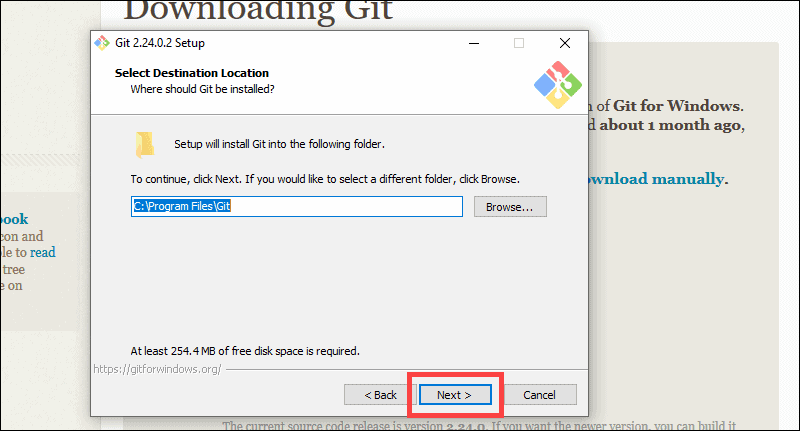
4. Allow the app to make changes to your device by clicking **Yes** on the User Account Control dialog that opens.



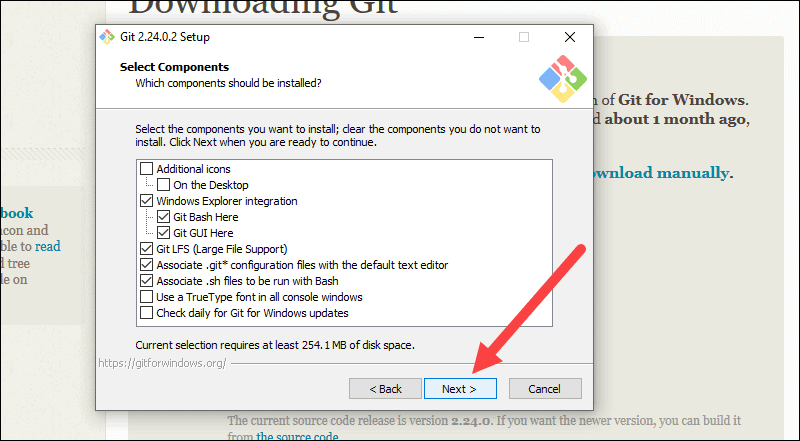
5. Review the GNU General Public License, and when you’re ready to install, click **Next**.



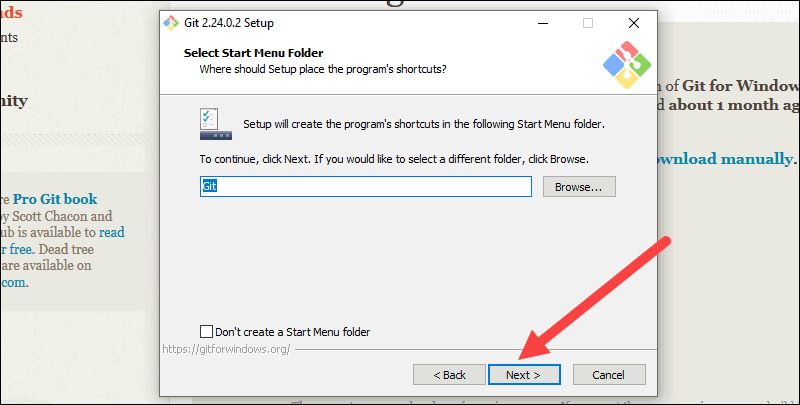
6. The installer will ask you for an installation location. Leave the default, unless you have reason to change it, and click **Next**.



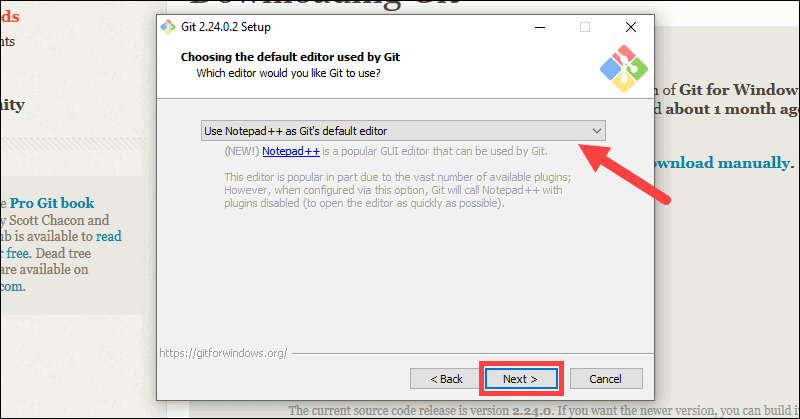
7. A component selection screen will appear. Leave the defaults unless you have a specific need to change them and click **Next**.



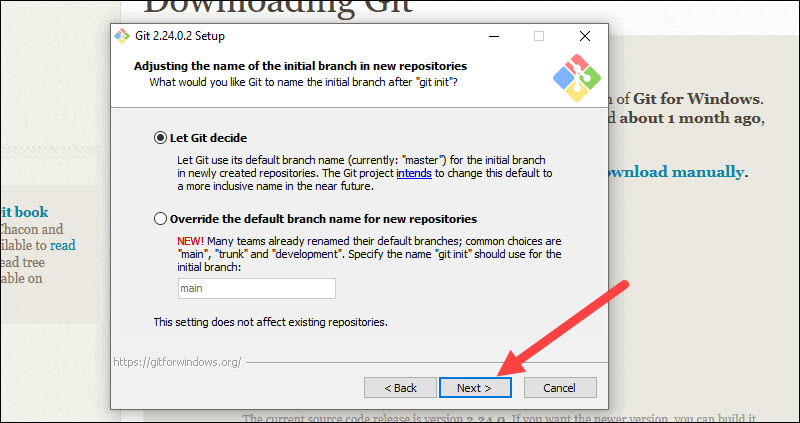
8. The installer will offer to create a start menu folder. Simply click **Next**.



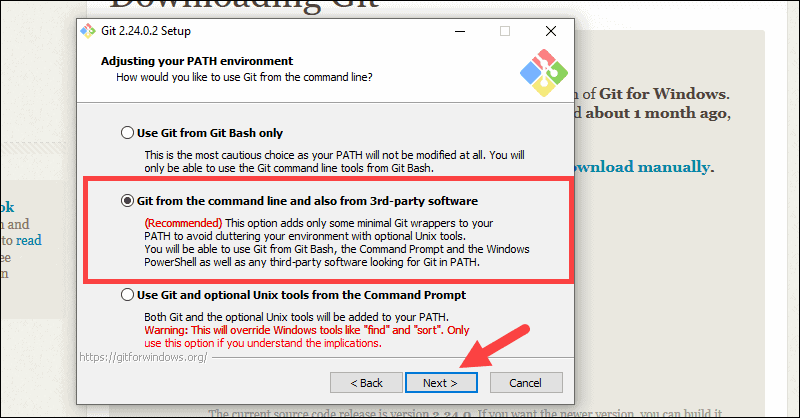
9. Select a text editor you’d like to use with Git. Use the drop-down menu to select Notepad++ (or whichever text editor you prefer) and click **Next**.



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.**

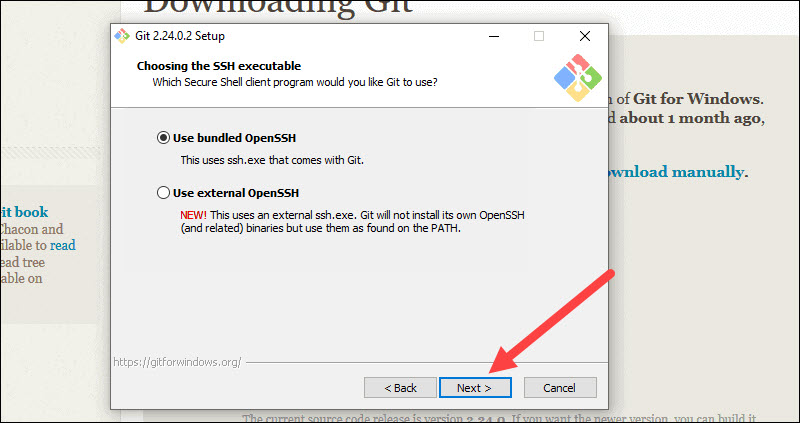


11. This installation step allows you to change the **PATH environment**. The **PATH**is the default set of directories included when you run a command from the command line. Leave this on the middle (recommended) selection and click **Next**.

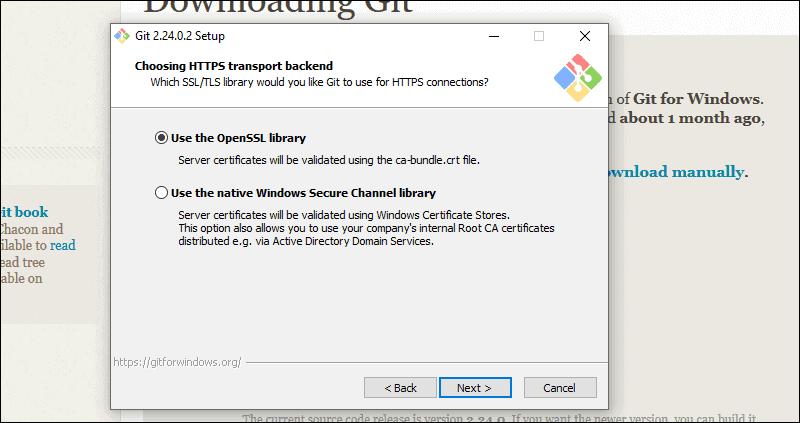


**Server Certificates, Line Endings and Terminal Emulators**

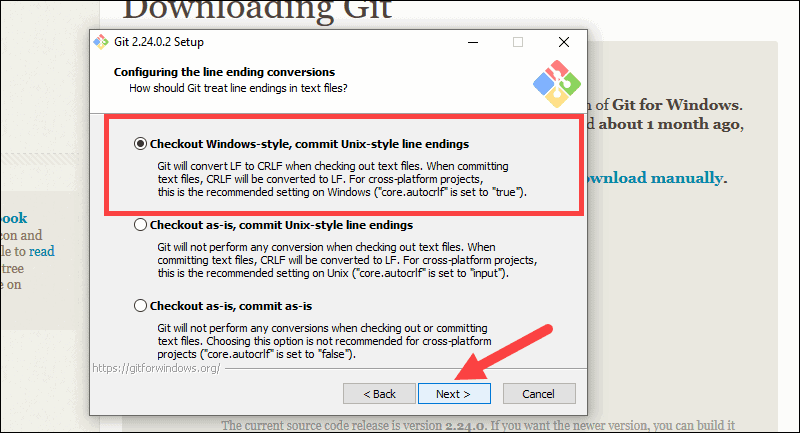
12. The installer now asks which SSH client you want Git to use. Git already comes with its own SSH client, so if you don't need a specific one, leave the default option and click **Next.**



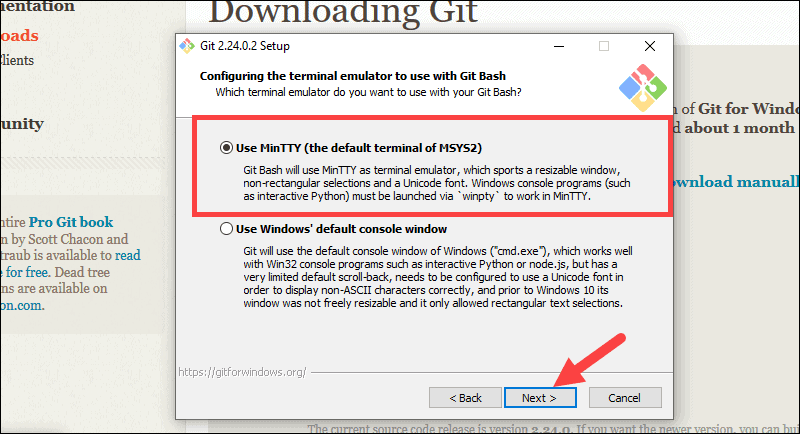
13. The next option relates to server certificates. Most users should use the default. If you’re working in an Active Directory environment, you may need to switch to Windows Store certificates. Click **Next**.



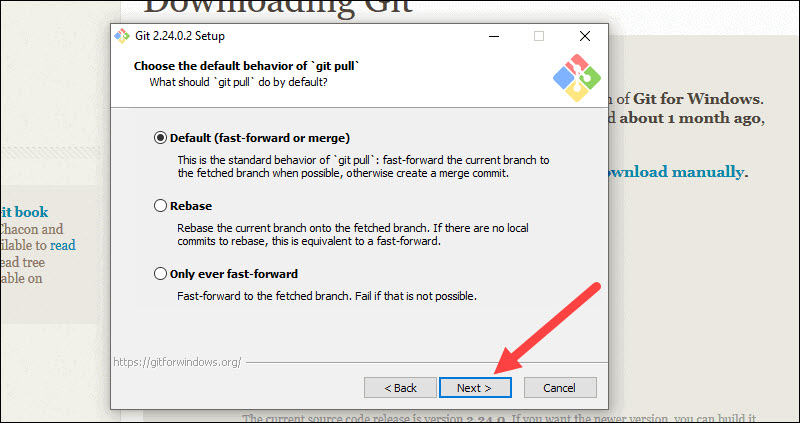
14. The next selection converts line endings. It is recommended that you leave the default selection. This relates to the way data is formatted and changing this option may cause problems. Click **Next**.



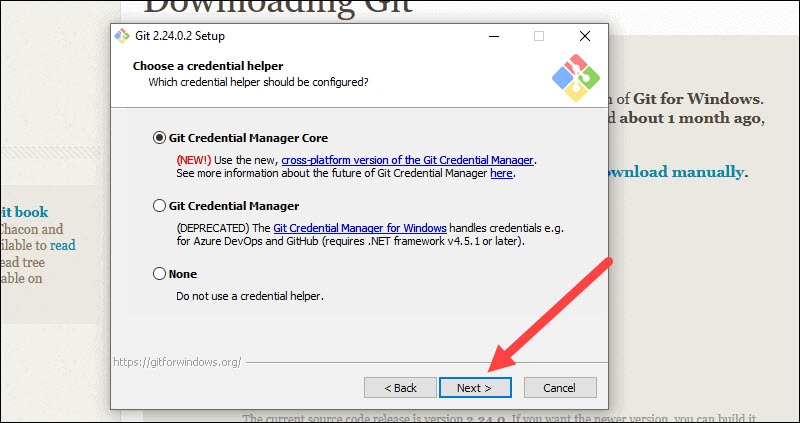
15. Choose the terminal emulator you want to use. The default MinTTY is recommended, for its features. Click **Next**.



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.

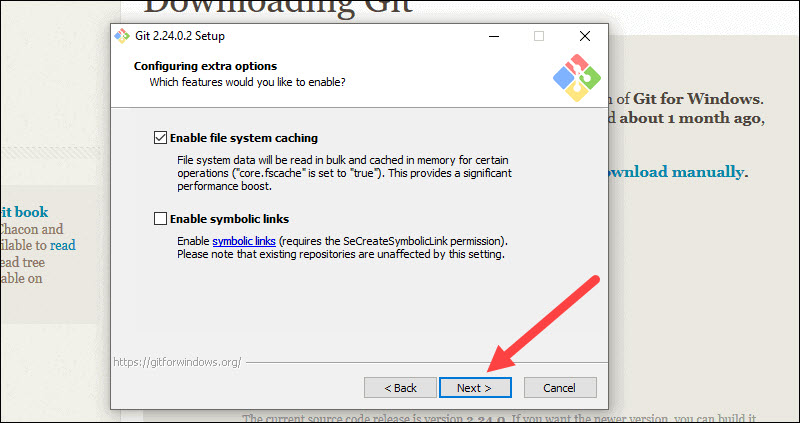


17. Next you should choose which credential helper to use. Git uses credential helpers to fetch or save credentials. Leave the default option as it is the most stable one, and click **Next**.

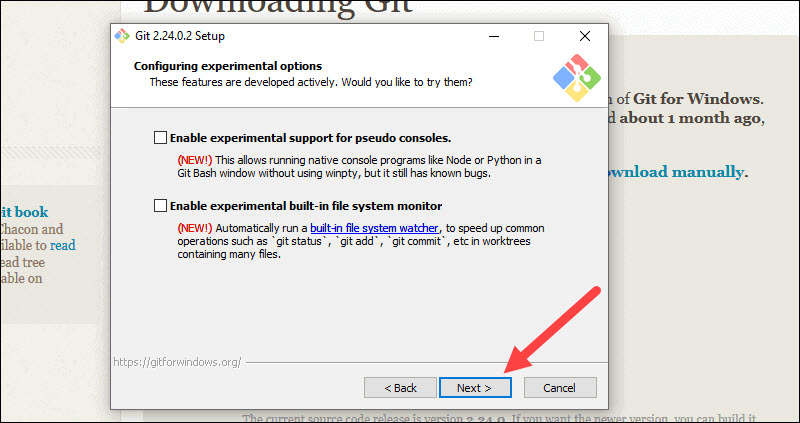


**Additional Customization Options**

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**.

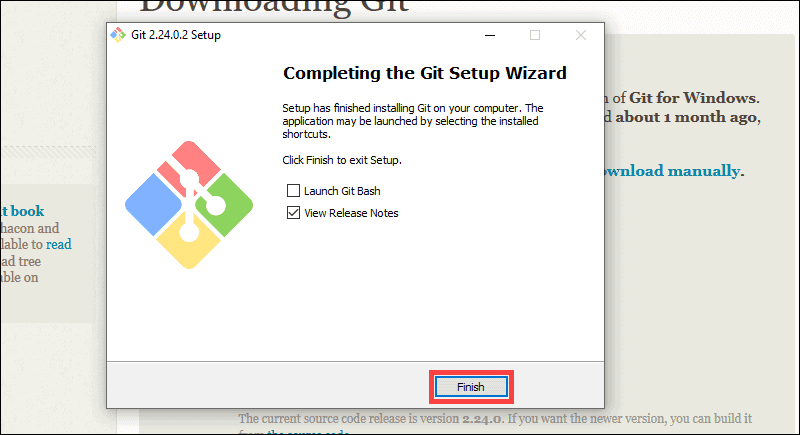


19. Depending on the version of Git you’re installing, it may offer to install experimental features. At the time this article was written, the options to include support for pseudo controls and a built-in file system monitor were offered. Unless you are feeling adventurous, leave them unchecked and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

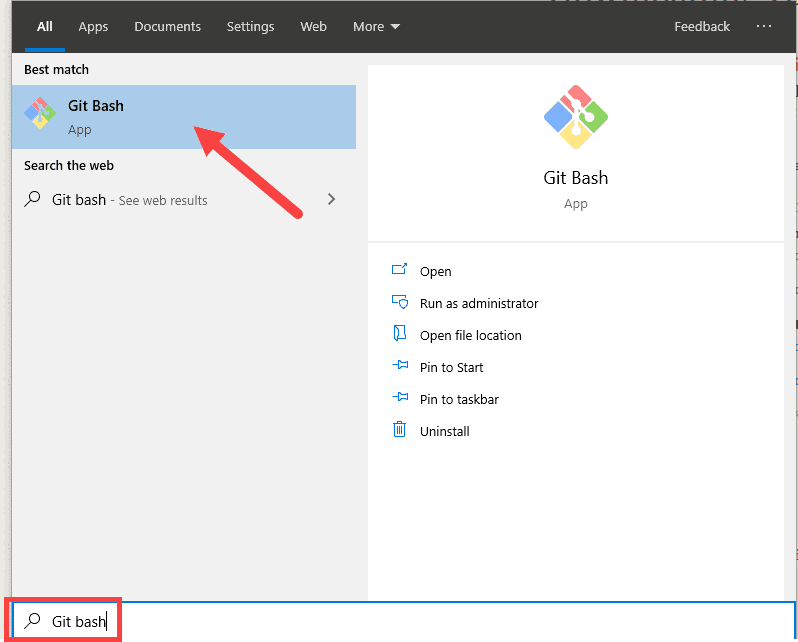


**How to Launch Git in Windows**

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

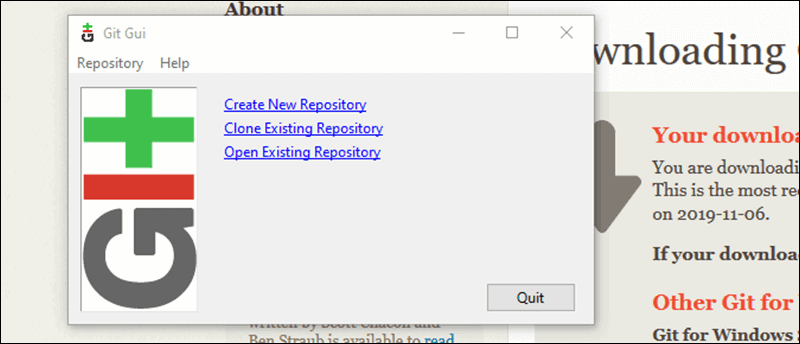
**Launch Git Bash Shell**

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.

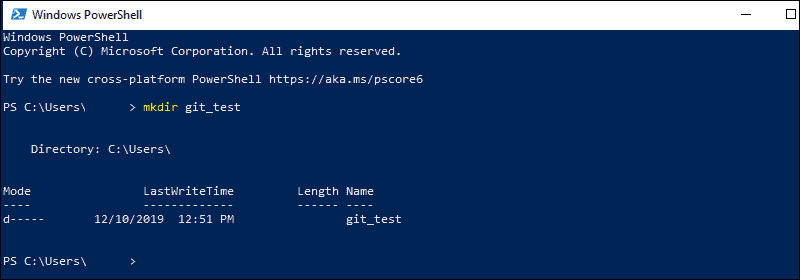
**Create a Test Directory**

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**.

Graphical user interface, text, application, email

Description automatically generated

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

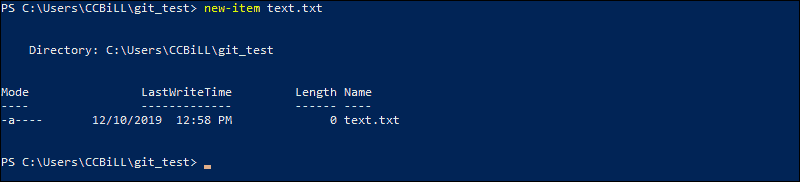
**Pushing Local Files to the Remote Repository**

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](https://phoenixnap.com/kb/git-create-new-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.