Git Local Environment Setup (Windows)

Read Time: 20 mins

In this article we will cover how to set up a local environment in order to use Git on your own Windows PC. Let's go through the steps and get you set up!

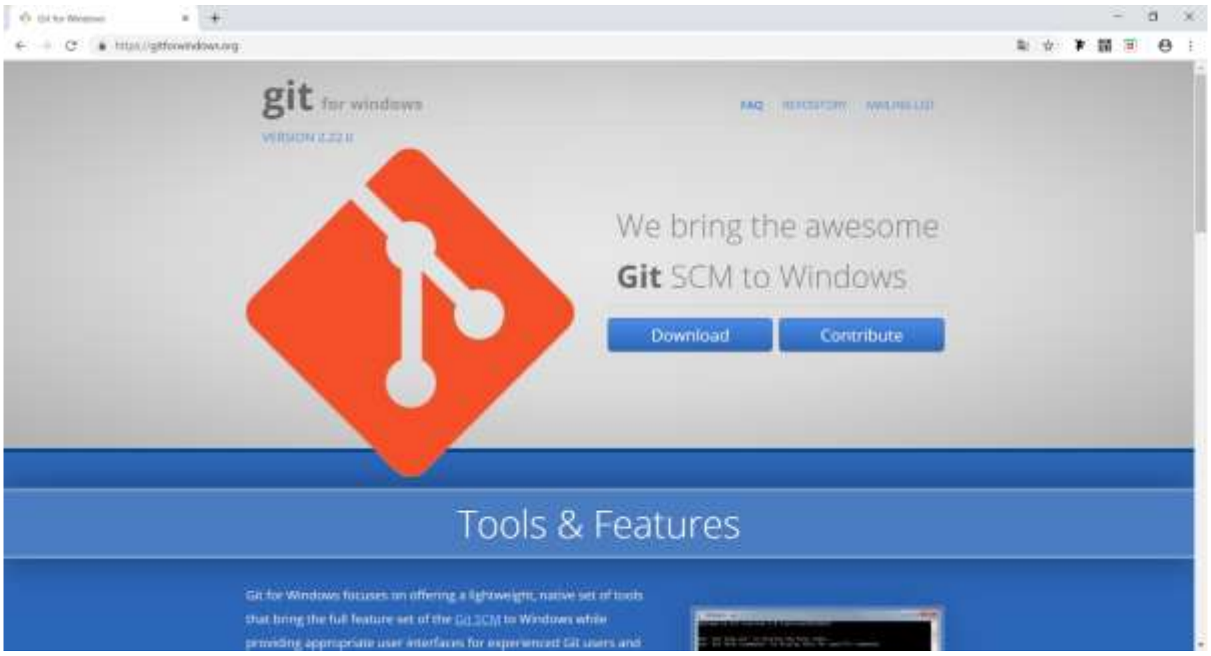
What You Need:

- Your own PC with Windows installed
- Basic knowledge to operate Command Prompt
- Text Editor (ie. Atom) (If you don't have Atom installed on your PC, please view the How to Make a Website with HTML and CSSarticle)
- To have completed the Progate Git Study I and Command Line Study I

Note: This article is for Windows users. Please read this article if you are using a Mac.

1. Installing Git

Firstly, let's install Git from Git for Windows.

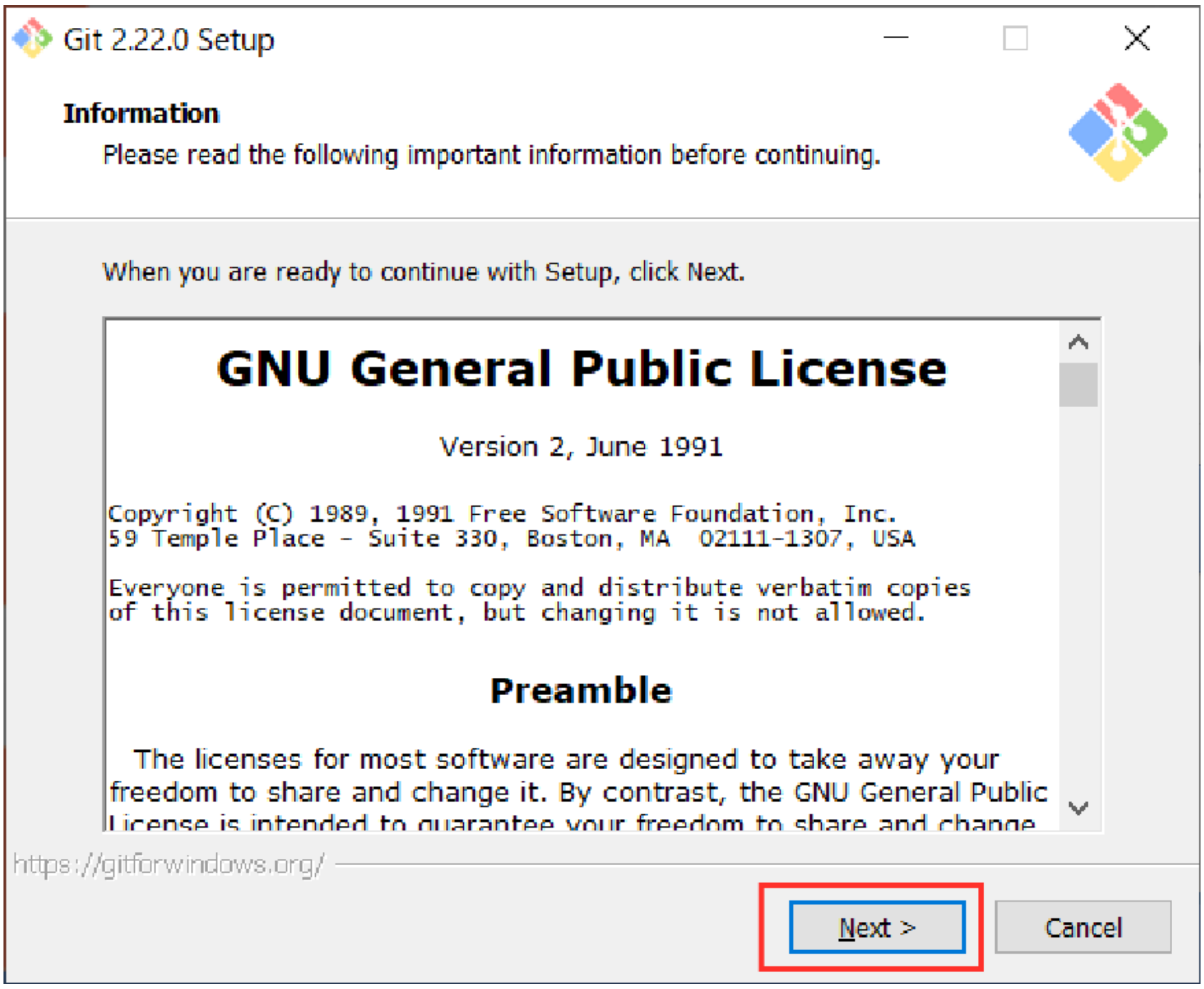


Git for Windows focuses on offering a lightweight, native set of tools that bring the full feature set of the Git SCM to Windows while providing appropriate user interfaces for experienced Git users and

Git for Windows

After the download is complete, open the file and an installation screen will pop up. Let's follow the prompts and complete the installation.

Step 1. Once you agree to the terms and conditions, click "Next"


Git 2.22.0 Setup

Information

Please read the following important information before continuing.

When you are ready to continue with Setup, click Next.

GNU General Public License

Version 2, June 1991

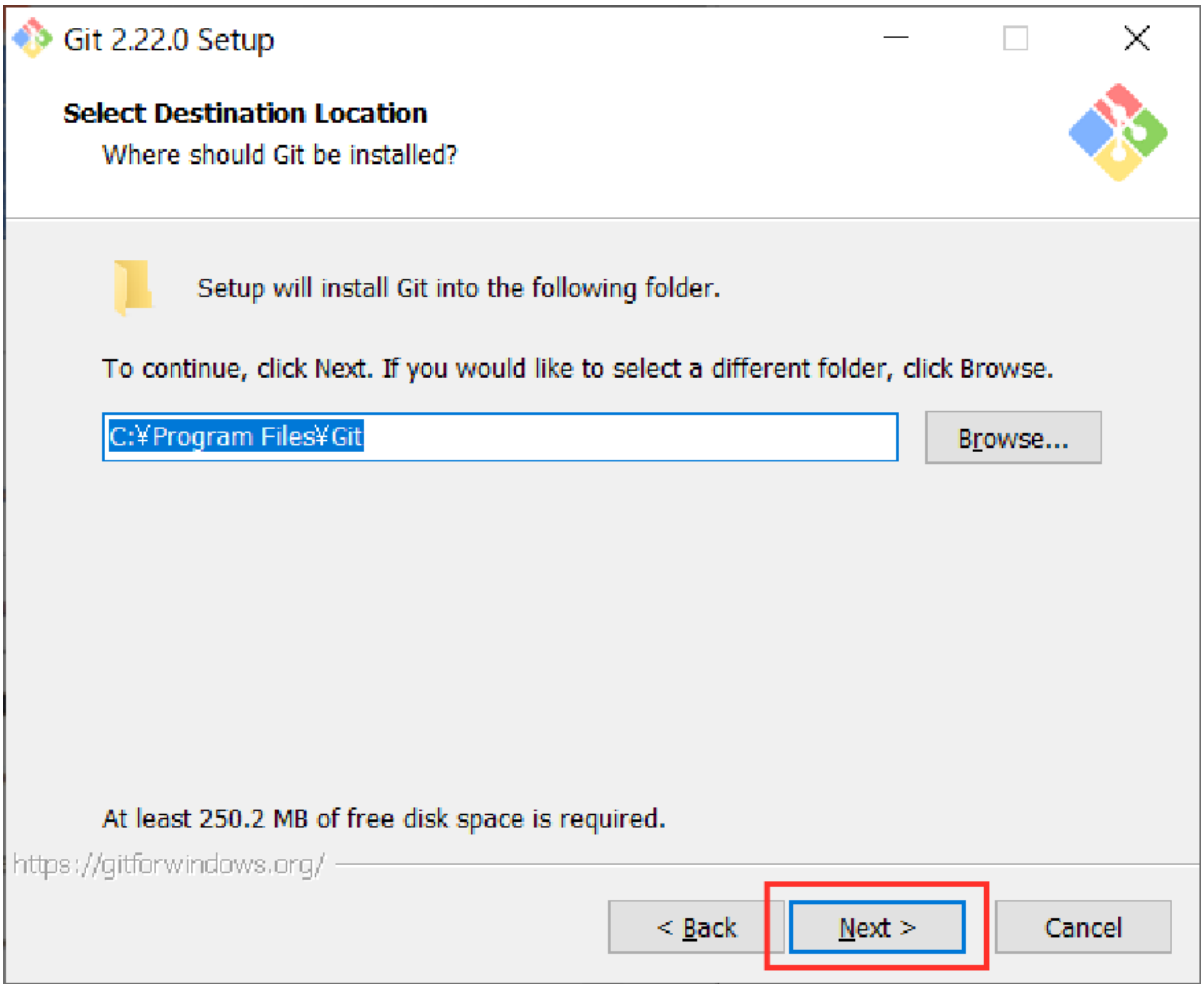
```
Copyright (C) 1989, 1991 Free Software Foundation, Inc.
59 Temple Place - Suite 330, Boston, MA  02111-1307, USA

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

Click "Next"

Step 2. Click "Next"


Git 2.22.0 Setup

Select Destination Location

Where should Git be installed?

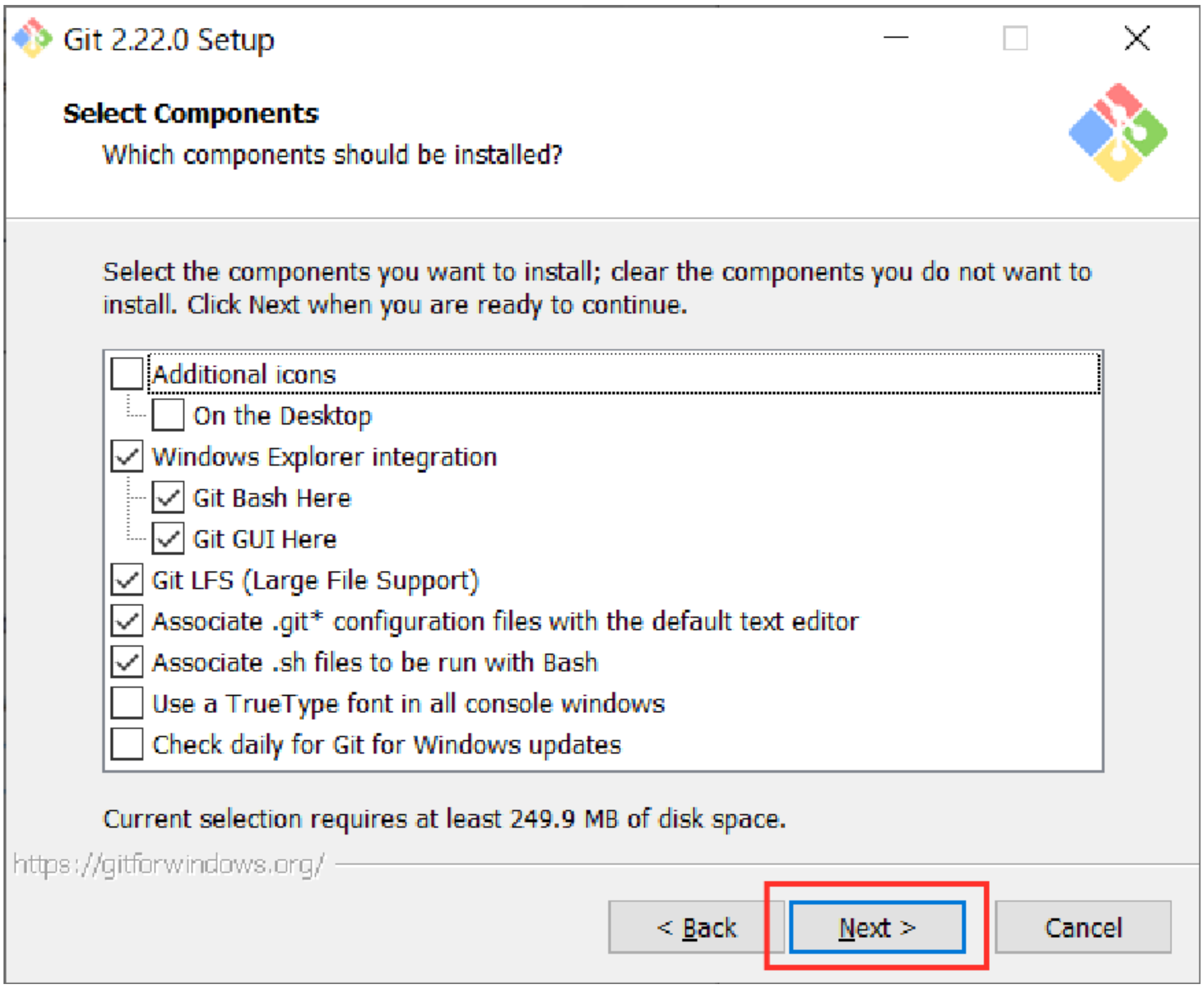
Setup will install Git into the following folder.

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

C:¥Program Files¥Git

Click "Next"

Step 3. Click "Next"


Git 2.22.0 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.

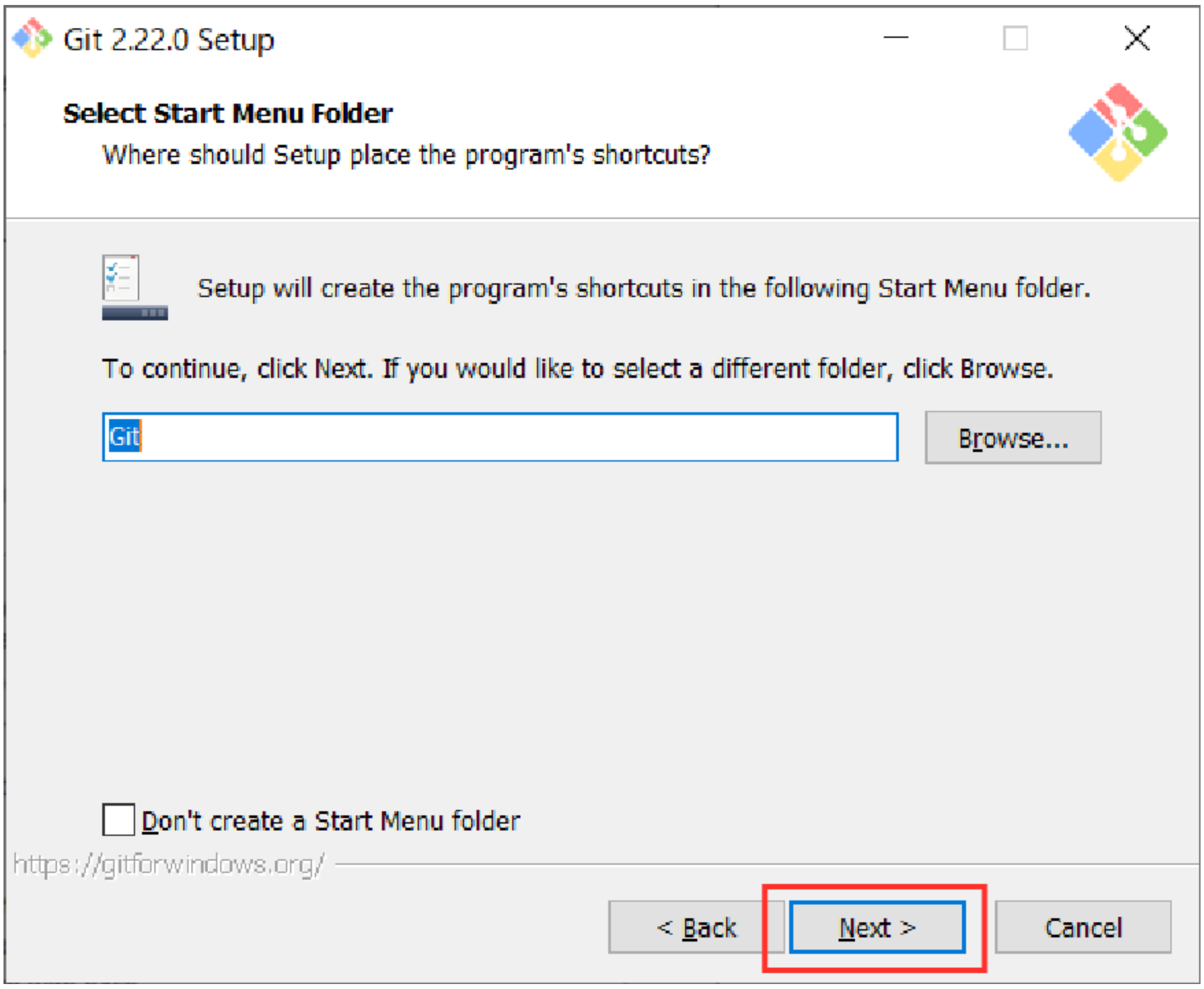
- [ ] Additional icons
  - [ ] On the Desktop
- [✓] Windows Explorer integration
  - [✓] Git Bash Here
  - [✓] Git GUI Here
- [✓] Git LFS (Large File Support)
- [✓] Associate .git* configuration files with the default text editor
- [✓] Associate .sh files to be run with Bash
- [ ] Use a TrueType font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

Current selection requires at least 249.9 MB of disk space.

https://gitforwindows.org/

Click "Next"

Step 4. Click "Next"


Git 2.22.0 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

Browse...

☐ Don't create a Start Menu folder

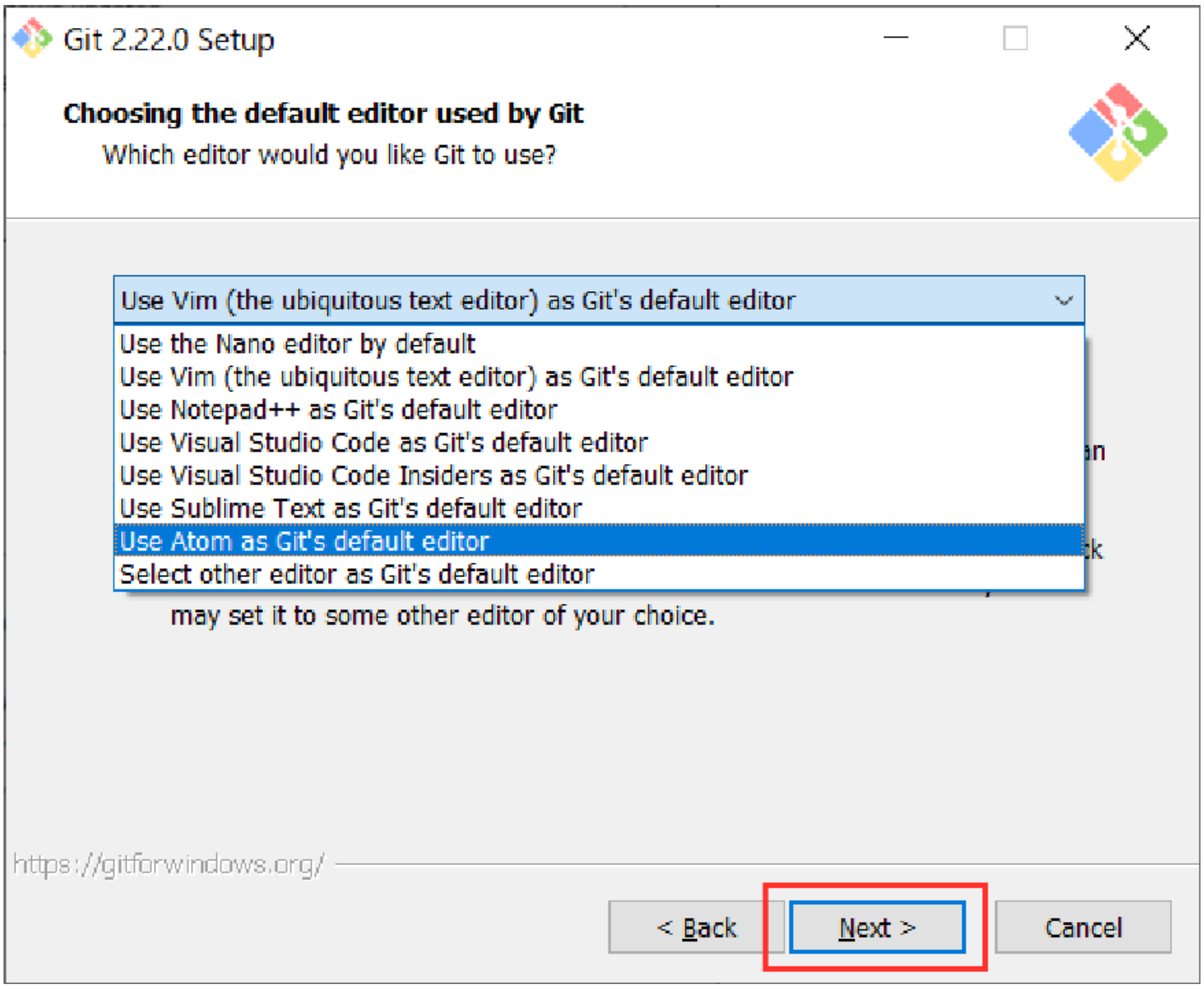
https://gitforwindows.org/

< Back | Next > | Cancel

Click "Next"

Step 5. We must select the editor. Choose the editor that you will use and click "Next"

(The example below is for Atom)



Git 2.22.0 Setup

Choosing the default editor used by Git

Which editor would you like Git to use?

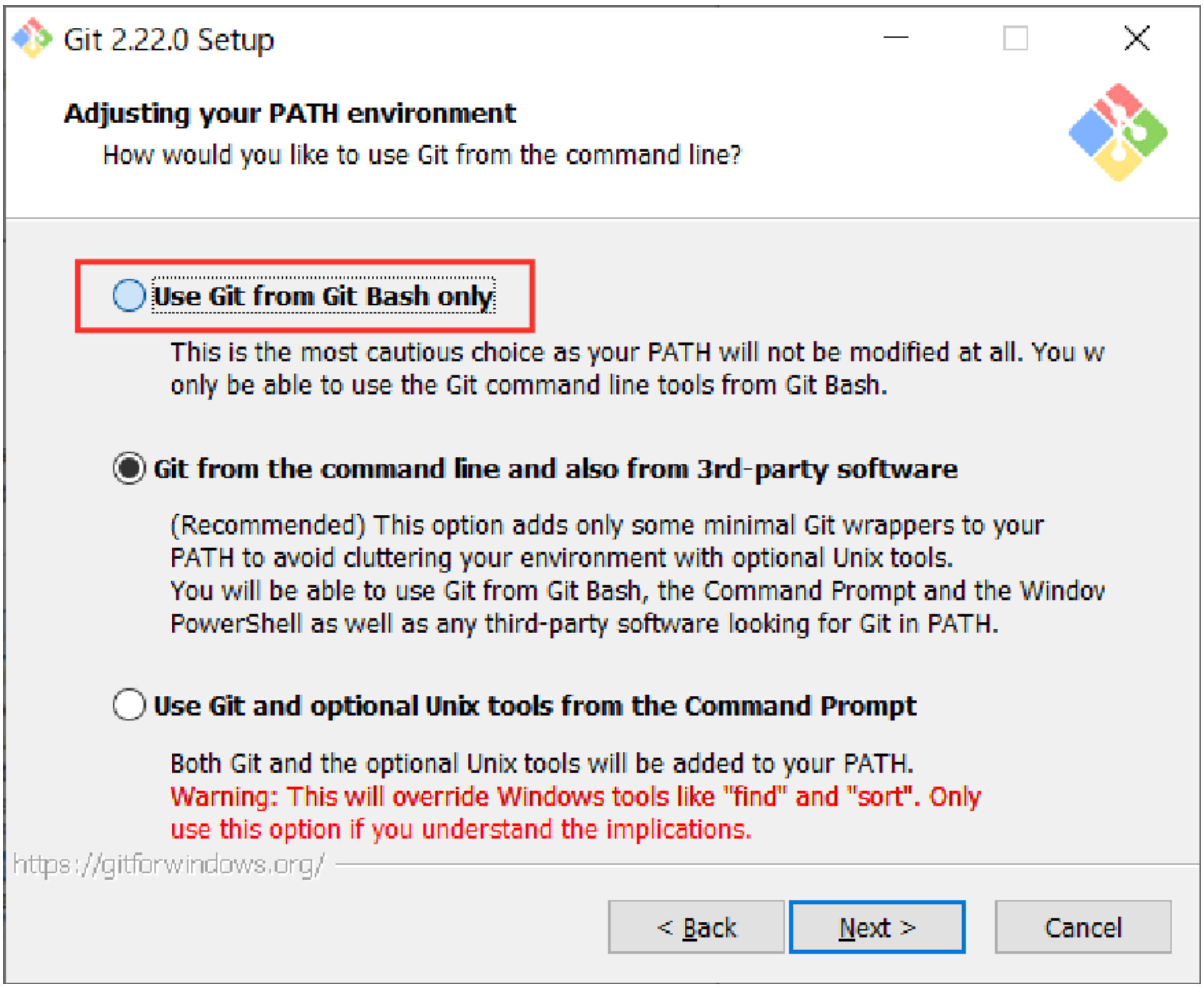
Use Vim (the ubiquitous text editor) as Git's default editor

- Use the Nano editor by default
- Use Vim (the ubiquitous text editor) as Git's default editor
- Use Notepad++ as Git's default editor
- Use Visual Studio Code as Git's default editor
- Use Visual Studio Code Insiders as Git's default editor
- Use Sublime Text as Git's default editor
- Use Atom as Git's default editor
- Select other editor as Git's default editor

may set it to some other editor of your choice.

Select the editor and click "Next"

Step 6. Check "Use Git from Git Bash only" and click "Next"



Git 2.22.0 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 w
only be able to use the Git command line tools from Git Bash.

(●) **Git from the command line and also from 3rd-party software**

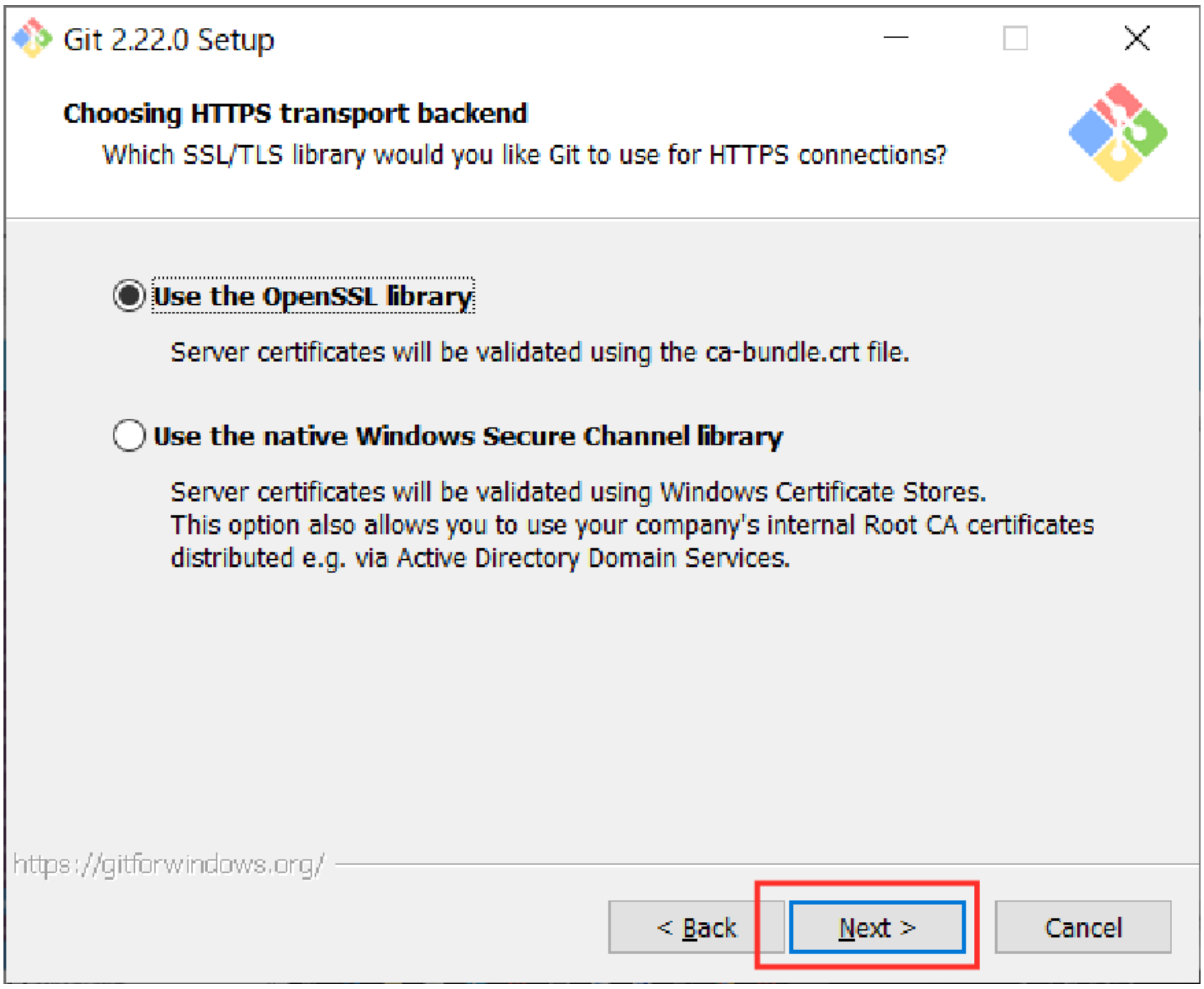
(Recommended) This option adds only some minimal Git wrappers to your
PATH to avoid cluttering your environment with optional Unix tools.
You will be able to use Git from Git Bash, the Command Prompt and the Windov
PowerShell as well as any third-party software looking for Git in PATH.

( ) **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.

https://gitforwindows.org/

Step 7. Check "Use the OpenSSL Library" and click "Next"


Git 2.22.0 Setup

Choosing HTTPS transport backend

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

[✓ **Use the OpenSSL library**]

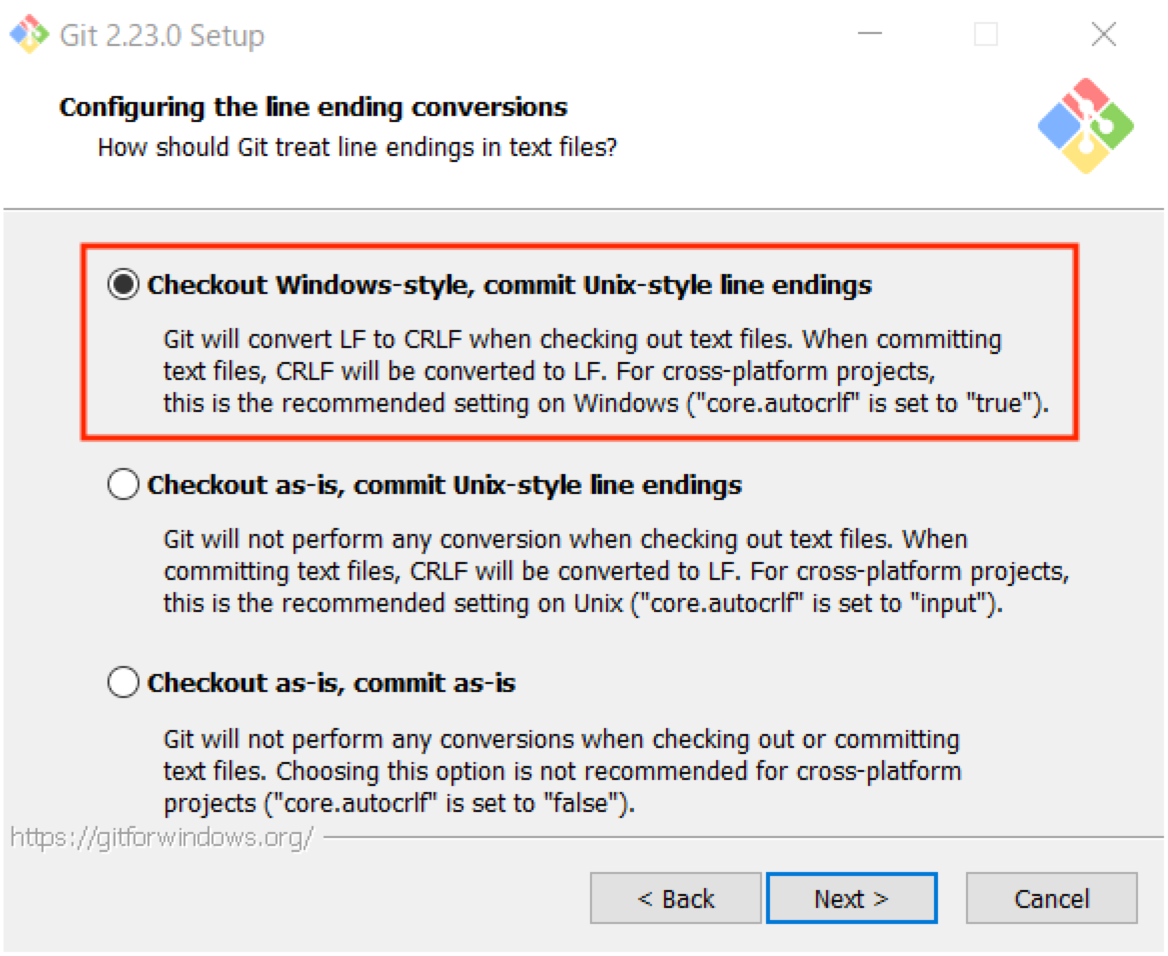
Server certificates will be validated using the ca-bundle.crt file.

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/

Step 8. Check "Checkout Windows-style, commit Unix-style line endings" and click "Next"


Git 2.23.0 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**

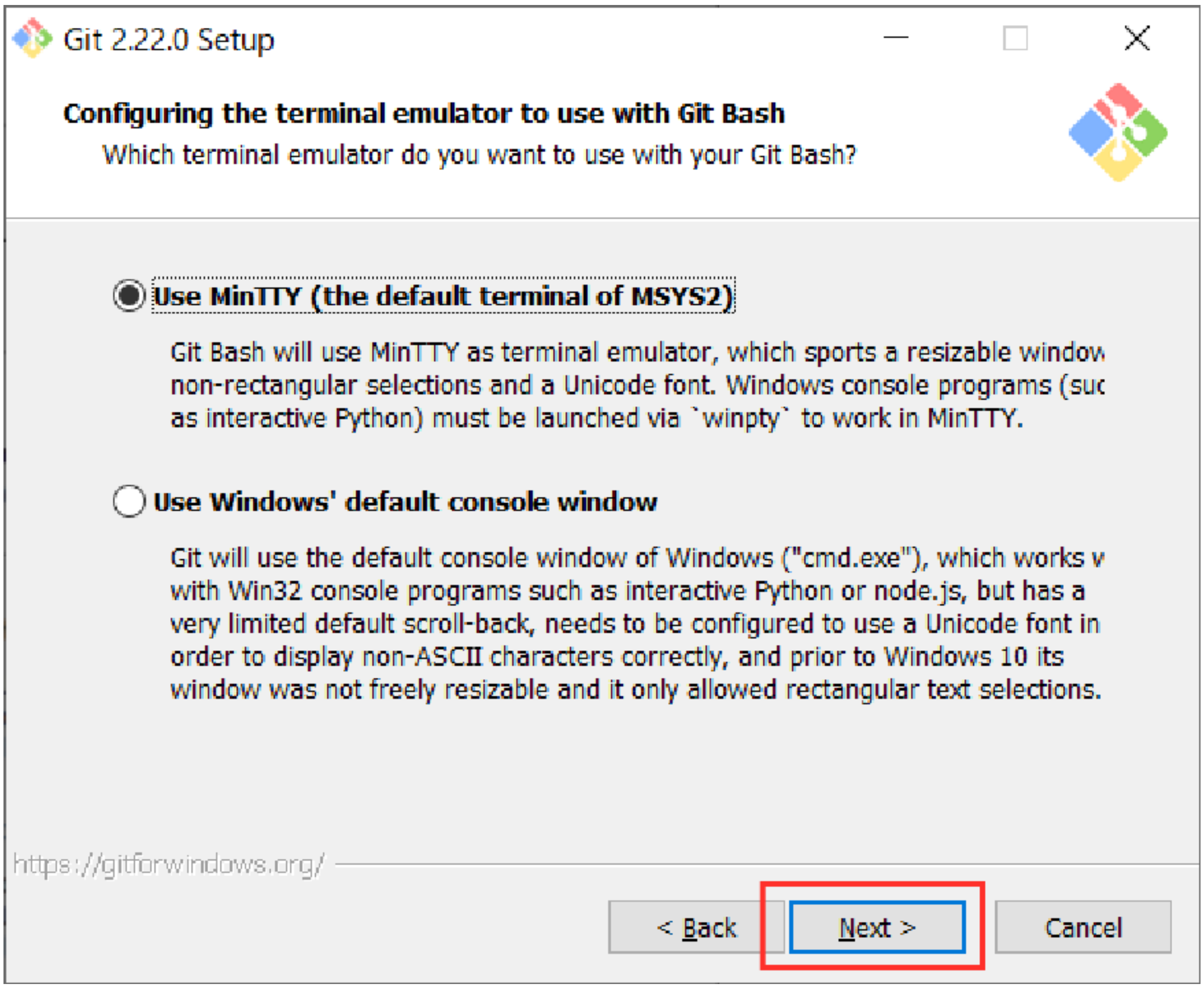
Git will not perform any conversion when checking out text files. When committing text files, CRLF will be converted to LF. For cross-platform projects, this is the recommended setting on Unix ("core.autocrlf" is set to "input").

( ) **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/

Step 9. Check "Use MinTTY (the default terminal of MSYS2)" and click "Next"


Git 2.22.0 Setup

Configuring the terminal emulator to use with Git Bash

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

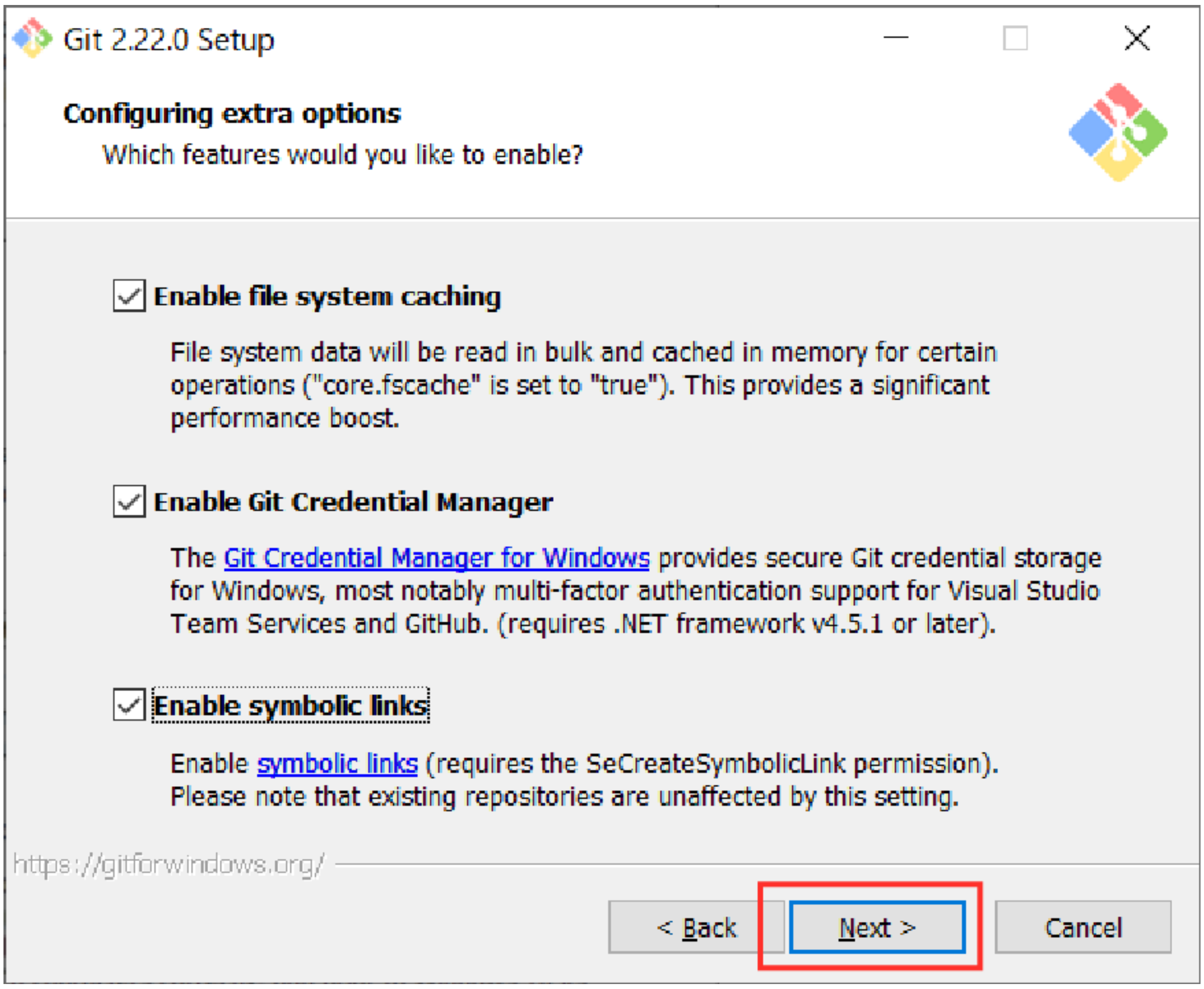
● **Use MinTTY (the default terminal of MSYS2)**

Git Bash will use MinTTY as terminal emulator, which sports a resizable window non-rectangular selections and a Unicode font. Windows console programs (suc 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 v 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.

Step 10. Check all of the options and click "Next"


Git 2.22.0 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.

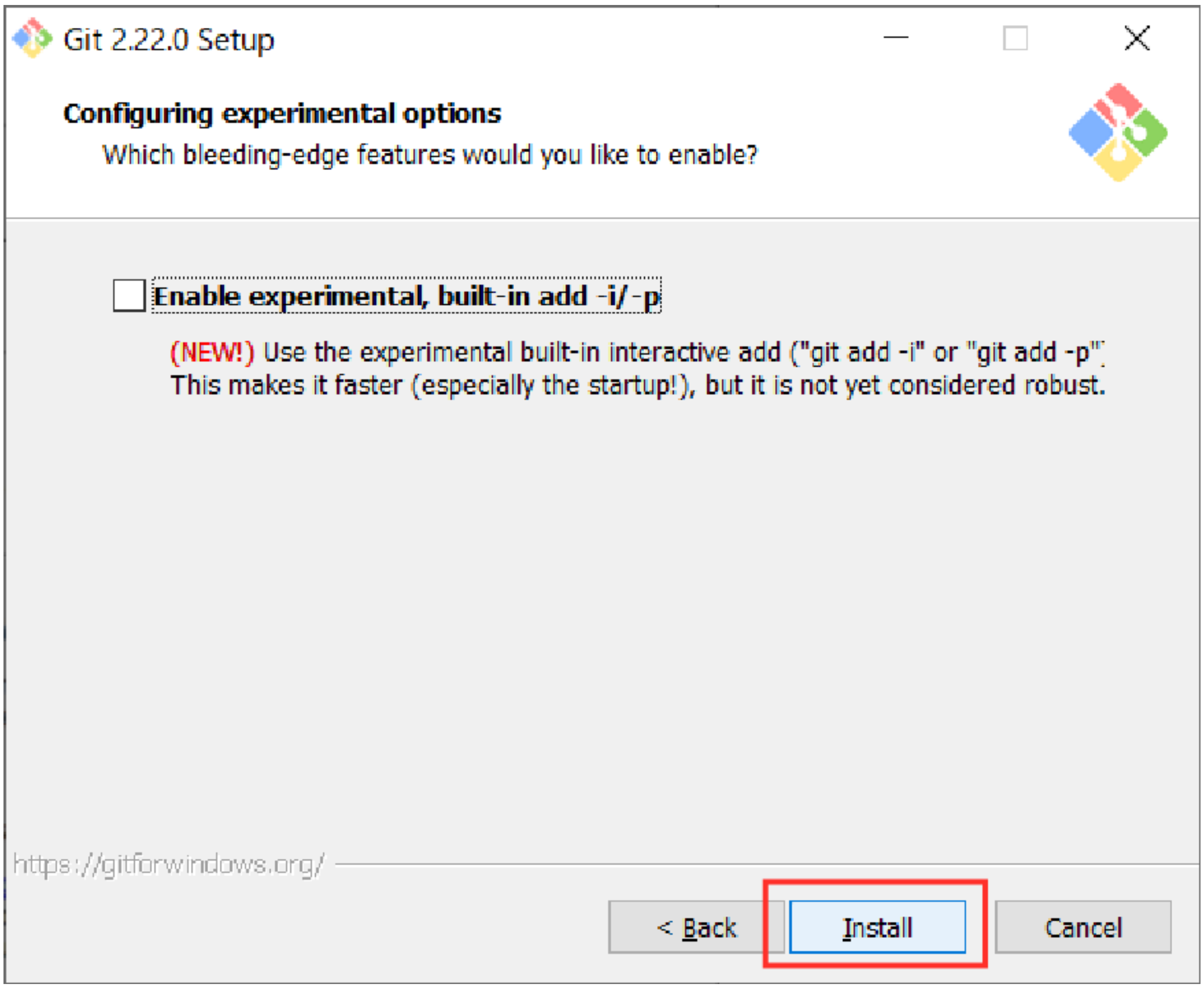
- [✓] **Enable Git Credential Manager**

  The Git Credential Manager for Windows provides secure Git credential storage for Windows, most notably multi-factor authentication support for Visual Studio Team Services and GitHub. (requires .NET framework v4.5.1 or later).

- [✓] **Enable symbolic links**

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

Step 11. Click "Install"


Git 2.22.0 Setup

Configuring experimental options

Which bleeding-edge features would you like to enable?

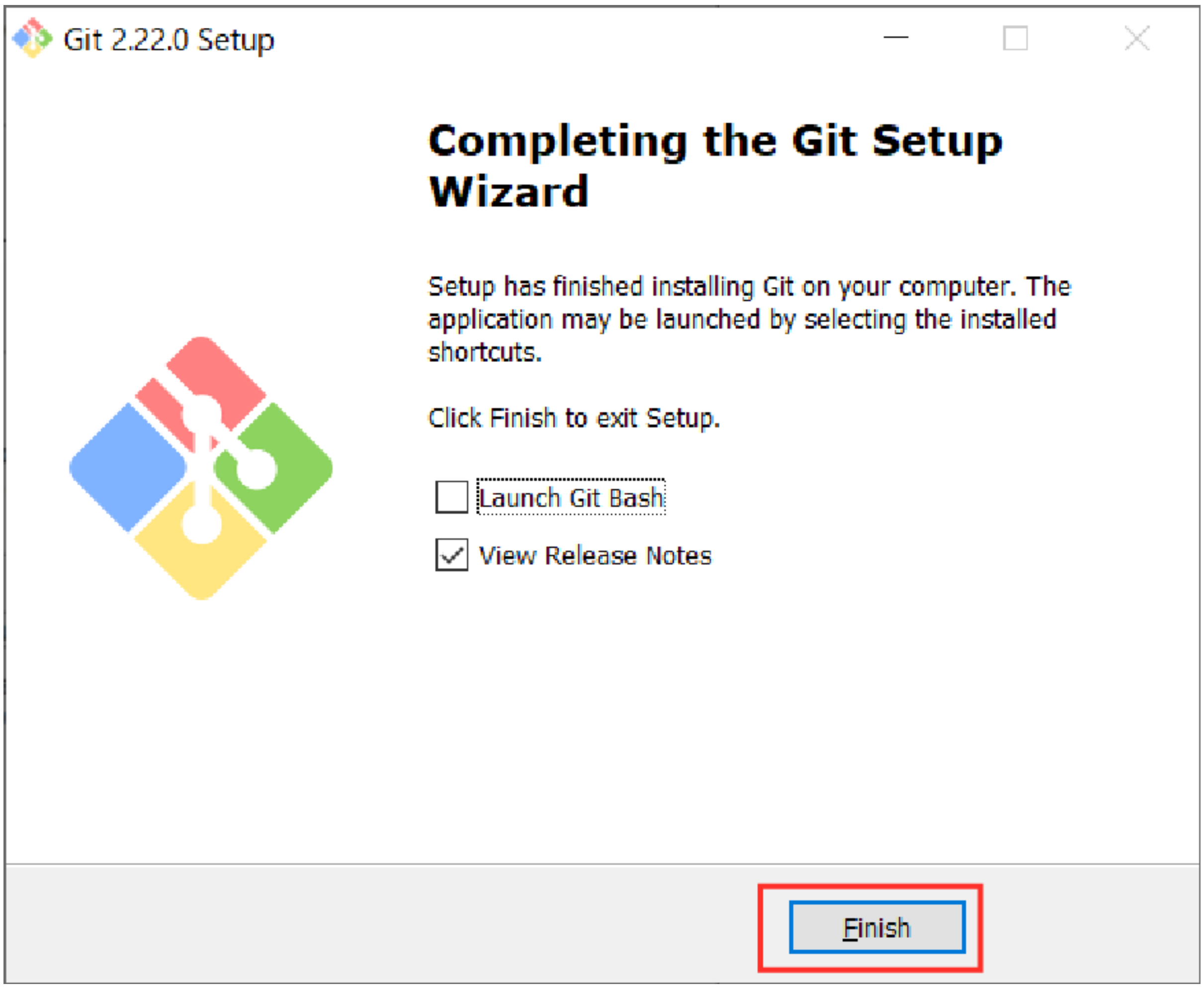
☐ **Enable experimental, built-in add -i/-p**

(NEW!) Use the experimental built-in interactive add ("git add -i" or "git add -p"
This makes it faster (especially the startup!), but it is not yet considered robust.

https://gitforwindows.org/

Now the installation will complete.

Step 12. Click "Finish" and you will be able to start using Git!



Git 2.22.0 Setup

Completing the Git Setup Wizard

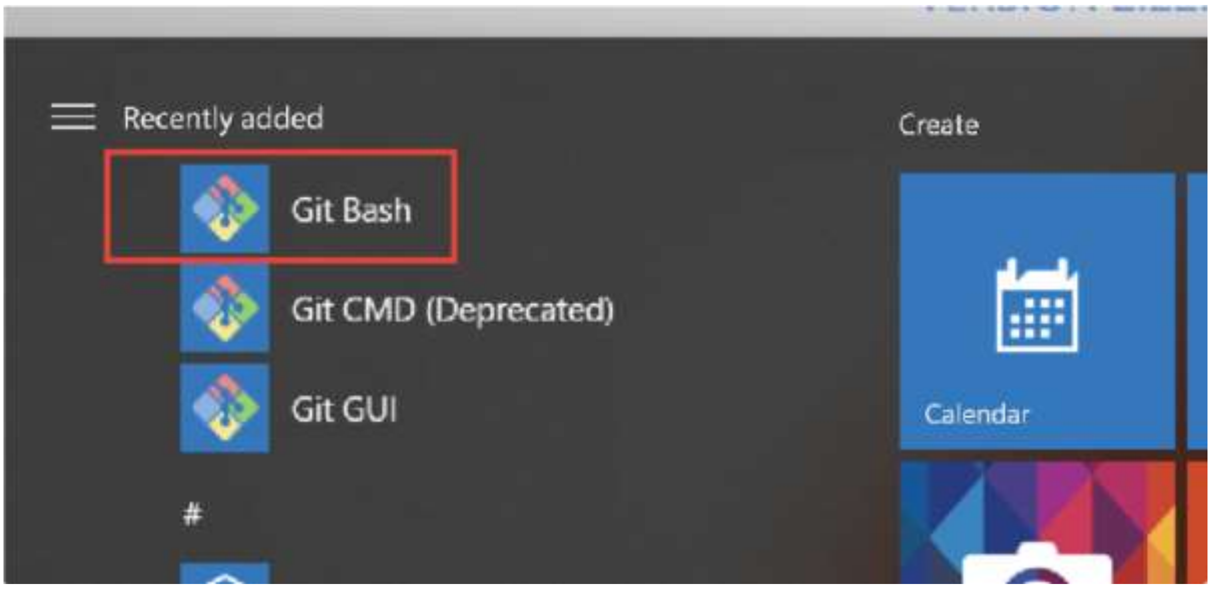
Setup has finished installing Git on your computer. The application may be launched by selecting the installed shortcuts.

Click Finish to exit Setup.

☐ Launch Git Bash

☑ View Release Notes

Now on your PC, there should be an application called **Git Bash** installed. This is a great tool to use for all your future development!



Git Bash has been installed

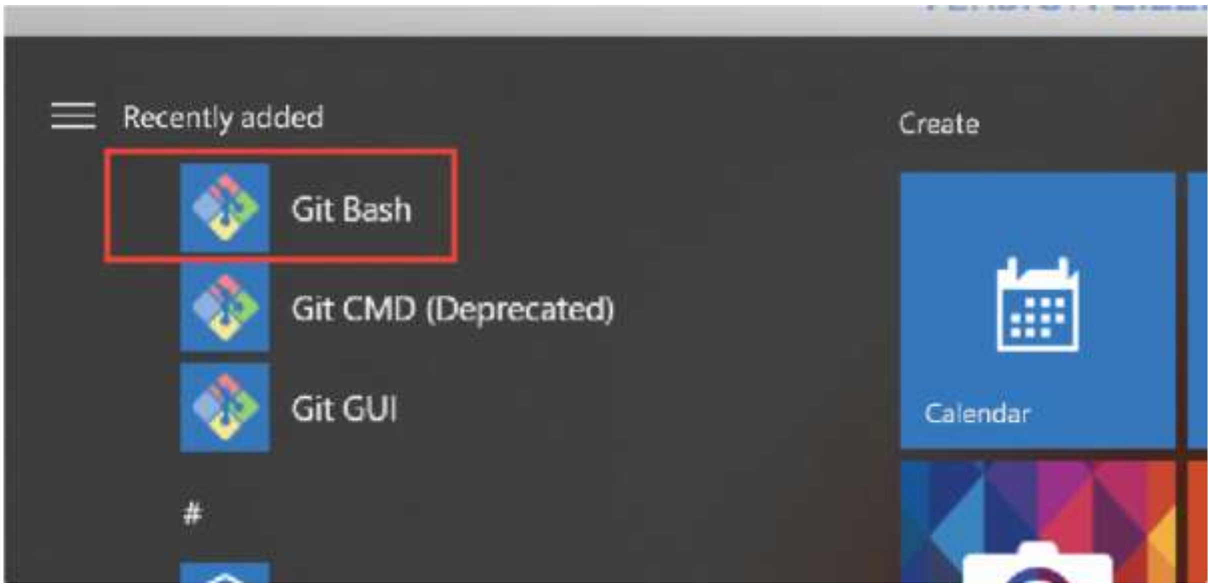
Check

Is Git installed on your PC?

Completed

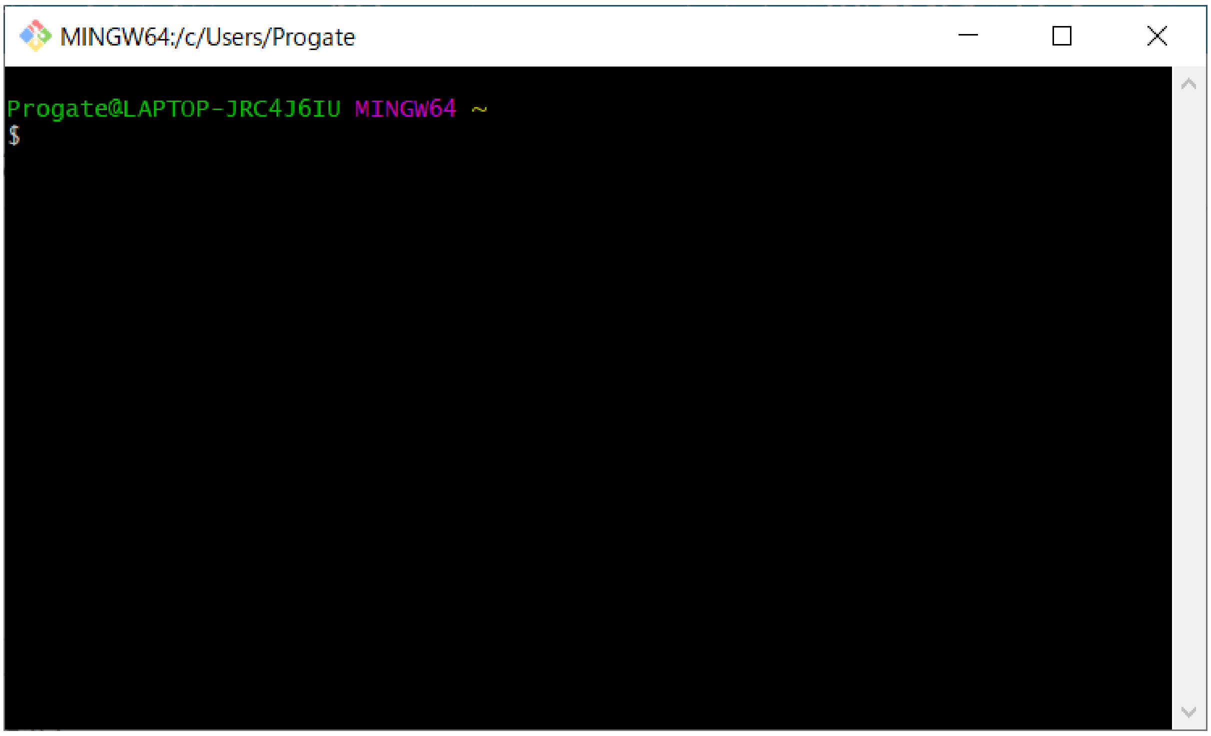
2. Initial Git Setup

First, let's open up **Git Bash**



Open up Git Bash

After it opens, your screen should look like the one below.


MINGW64:/c/Users/Progate

```
Progate@LAPTOP-JRC4J6IU MINGW64 ~
$
```

Git Bash startup screen

Set up a username and email address

By setting up a username and email address in Git, any commits made will be recorded and you will be able to identify who the commits were created by.

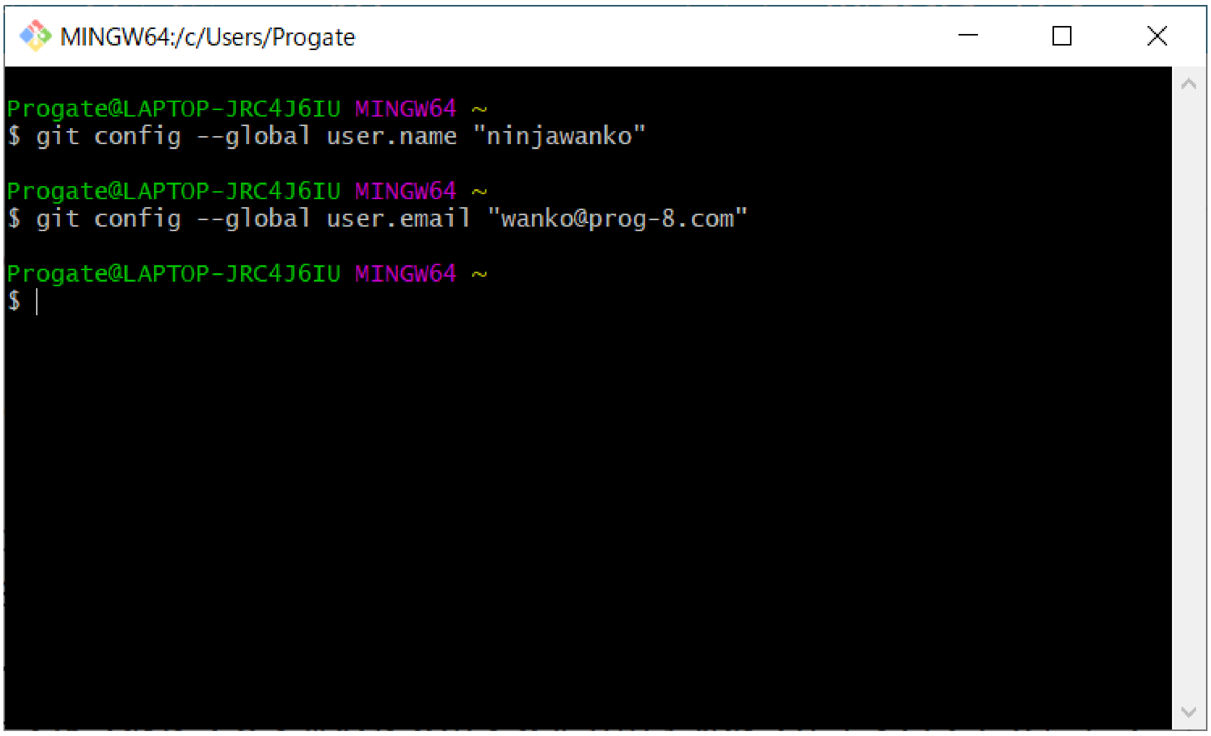
Run the commands below to set up your details:

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

```
git config --global user.email "email address"
```

(You can choose your own username and email address)

Below is just an example.



```
Progate@LAPTOP-JRC4J6IU MINGW64 ~
$ git config --global user.name "ninjawanko"

Progate@LAPTOP-JRC4J6IU MINGW64 ~
$ git config --global user.email "wanko@prog-8.com"

Progate@LAPTOP-JRC4J6IU MINGW64 ~
$ |
```

Setting up the user details

Now the initial setup is complete!

Check

Have you successfully setup username and email address?

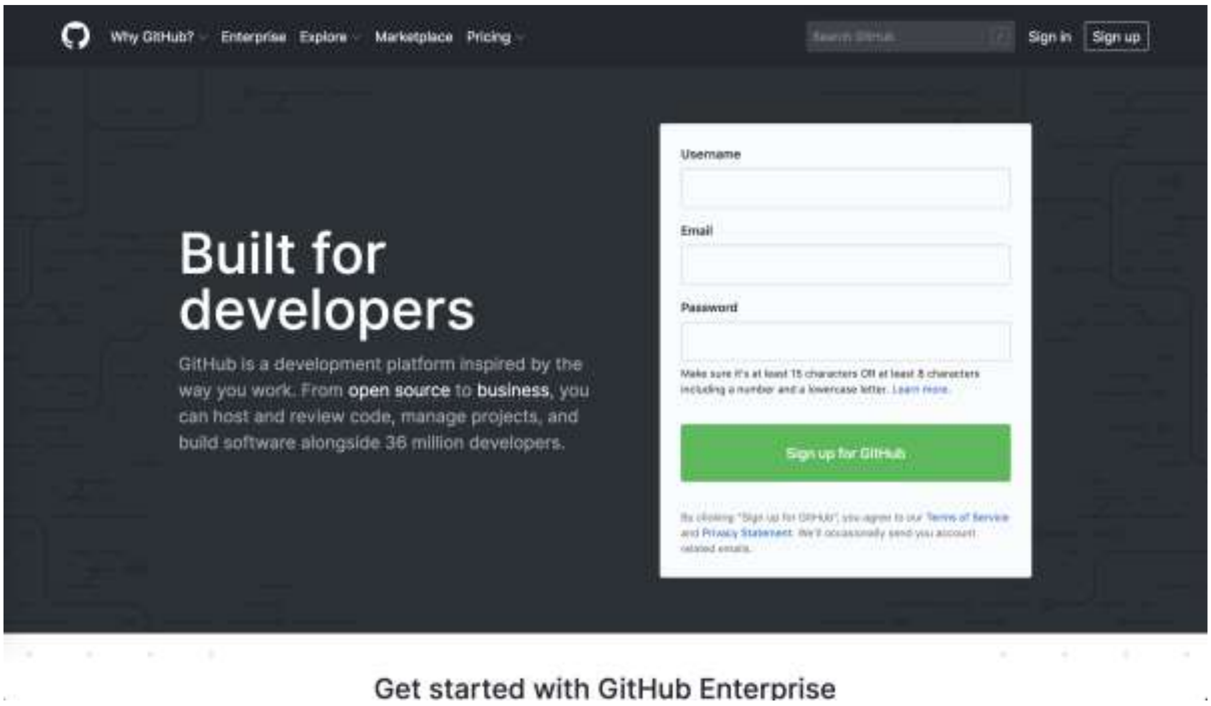
Completed

3. Setting Up GitHub

From here, we will be looking at how to add a new remote repository by utilizing a service called Github.
You will create a remote repository like the one Master Wooly prepared for you in **Git Study I**.

Setting up a Github account

In order to utilize Github, let's make an account.
To create an account, visit the Github Official Page



Built for developers

GitHub is a development platform inspired by the way you work. From **open source** to **business**, you can host and review code, manage projects, and build software alongside 36 million developers.

Username

Email

Password

Make sure it's at least 15 characters OR at least 8 characters including a number and a lowercase letter. Learn more.

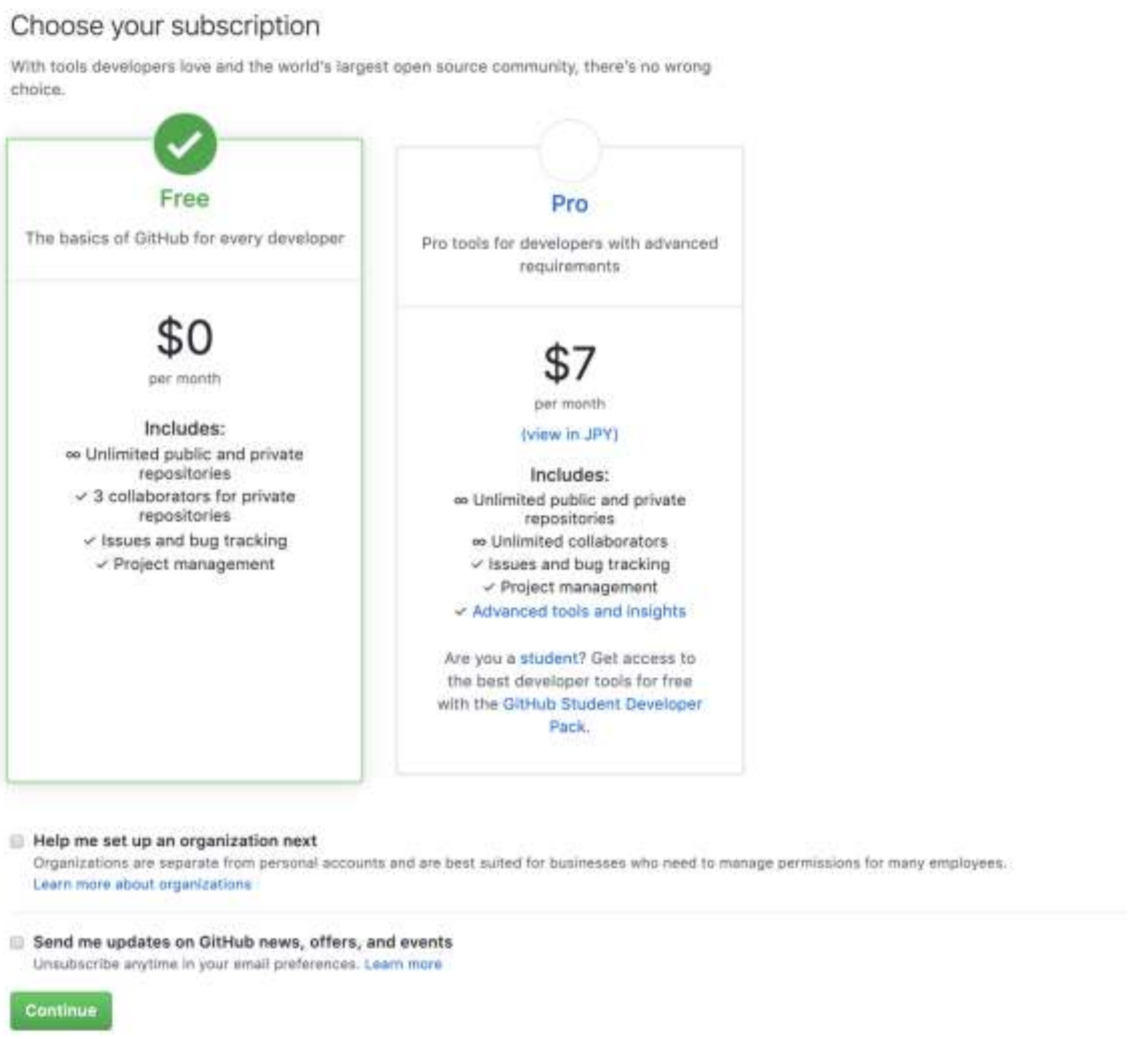
By clicking "Sign up for GitHub", you agree to our Terms of Service and Privacy Statement. We'll occasionally send you account related emails.

Get started with GitHub Enterprise

Screen when you access the page

Once you have signed up and followed the prompts, you will be taken to a subscription selection page.

To use Github for free, click the "Free" option and then the "Continue" button at the bottom left.



Choose your subscription

With tools developers love and the world's largest open source community, there's no wrong choice.

Free

The basics of GitHub for every developer

$0
per month

Includes:
- ∞ Unlimited public and private repositories
- ✓ 3 collaborators for private repositories
- ✓ Issues and bug tracking
- ✓ Project management

Pro

Pro tools for developers with advanced requirements

$7
per month
(view in JPY)

Includes:
- ∞ Unlimited public and private repositories
- ∞ Unlimited collaborators
- ✓ Issues and bug tracking
- ✓ Project management
- ✓ Advanced tools and insights

Are you a student? Get access to the best developer tools for free with the GitHub Student Developer Pack.

☐ **Help me set up an organization next**
Organizations are separate from personal accounts and are best suited for businesses who need to manage permissions for many employees.
Learn more about organizations

☐ **Send me updates on GitHub news, offers, and events**
Unsubscribe anytime in your email preferences. Learn more

Subscription option page

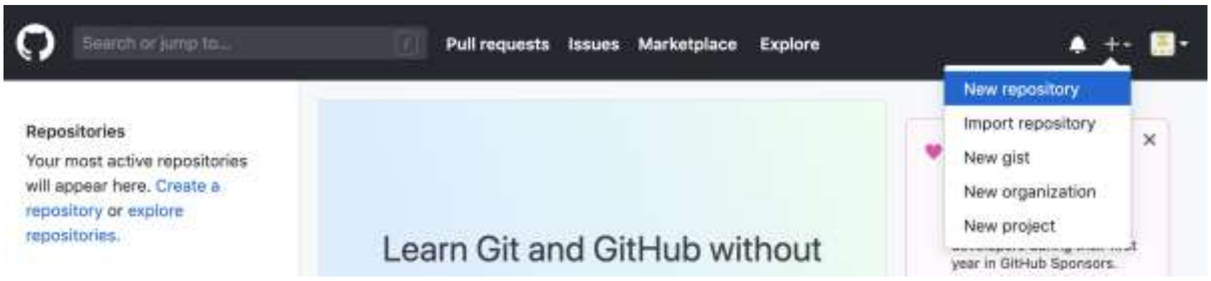
After selecting the subscription you will be asked multiple questions, but you do not need to answer them.

Lastly, click the "Submit" button and your account setup will be complete!

Once you have set up your account, you will receive a verification email. Click on the link attached in the email, and now you will be able to start using your Github account.

Creating a new remote repository

To create a new remote repository, click on the "+" button in the header on the top right.

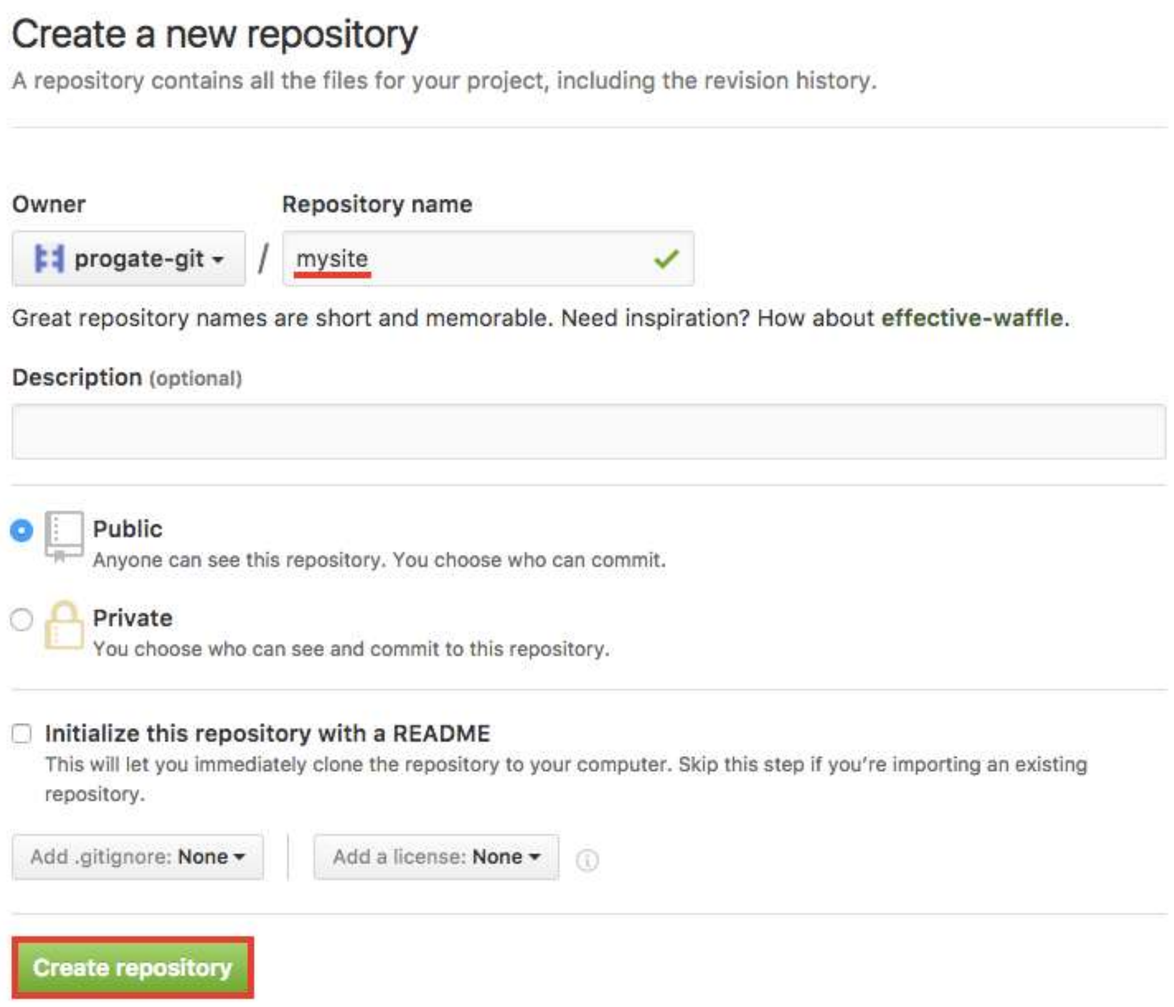


Creating a new remote repository

After clicking the "+" button, click on "New repository."

By clicking "New repository", you will be navigated to the page below.

Type in a project name where it says "Repository name" and click "Create repository" to create a new remote repository.



Create a new repository

A repository contains all the files for your project, including the revision history.

Owner: progate-git / Repository name: mysite ✓

Great repository names are short and memorable. Need inspiration? How about **effective-waffle**.

Description (optional)

● **Public**
Anyone can see this repository. You choose who can commit.

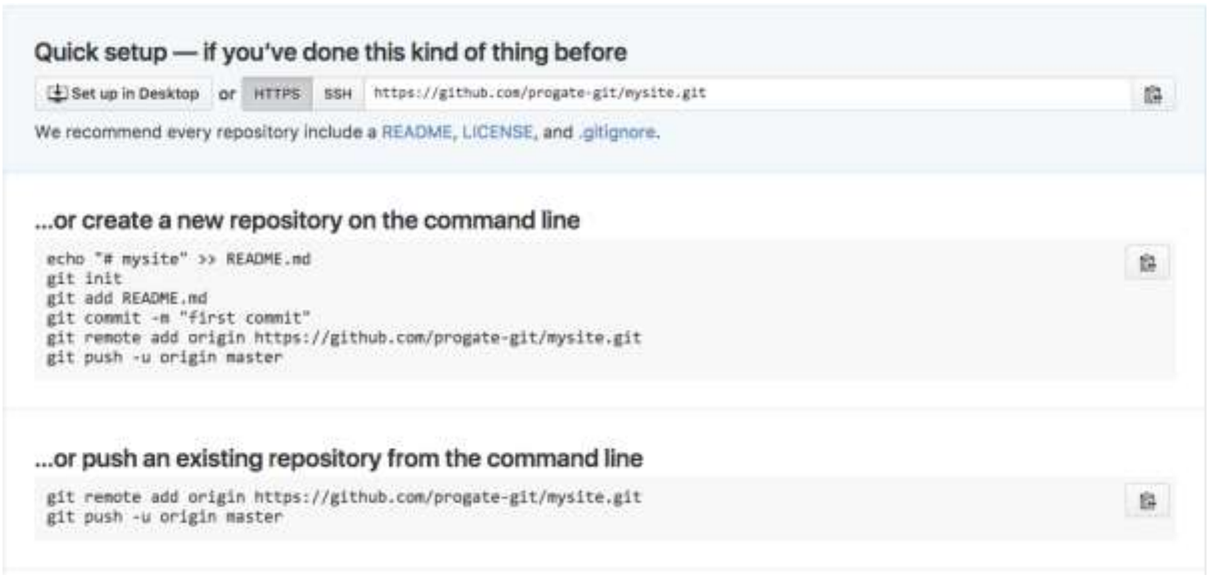
○ **Private**
You choose who can see and commit to this repository.

☐ **Initialize this repository with a README**
This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: **None** | Add a license: **None**

New repository details

Once the remote repository is created, you will be directed to the page below.


Quick setup — if you've done this kind of thing before

```
https://github.com/progate-git/mysite.git
```

We recommend every repository include a README, LICENSE, and .gitignore.

...or create a new repository on the command line

```
echo "# mysite" >> README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin https://github.com/progate-git/mysite.git
git push -u origin master
```

...or push an existing repository from the command line

```
git remote add origin https://github.com/progate-git/mysite.git
git push -u origin master
```

The remote repository has been created

Check

Were you able to register GitHub and create a new remote repository?

Completed

4. Pushing to GitHub

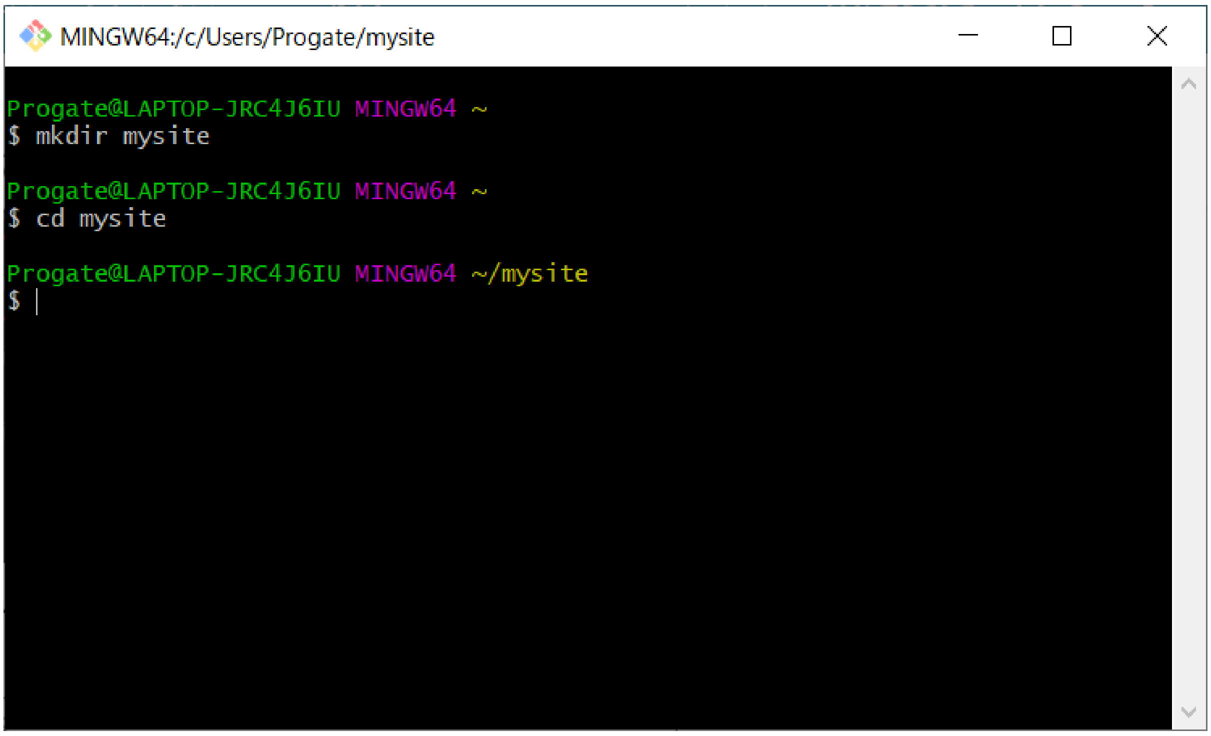
Let's try pushing local repository content to the remote repository we created on GitHub.

Preparing to use Git

Using Git, you must first create a directory and move to that directory by running the following commands:

```
mkdir mysite
```

```
cd mysite
```


MINGW64:/c/Users/Progate/mysite

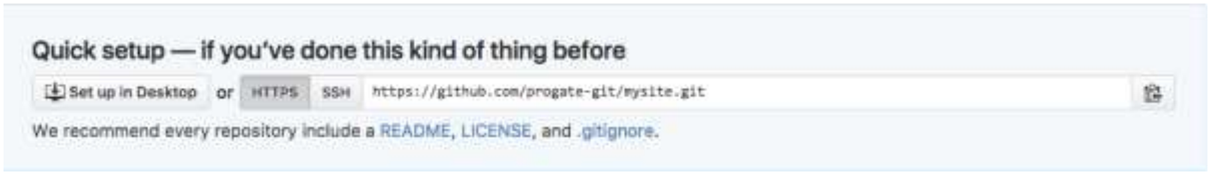
```
Progate@LAPTOP-JRC4J6IU MINGW64 ~
$ mkdir mysite

Progate@LAPTOP-JRC4J6IU MINGW64 ~
$ cd mysite

Progate@LAPTOP-JRC4J6IU MINGW64 ~/mysite
$ |
```

After running the commands

To be able to manage that file in Git, you must run the following command:

```
git init
```

After you have run the `git init` command, please copy the line starting with "https" as shown below.


Quick setup — if you've done this kind of thing before

Set up in Desktop or HTTPS SSH https://github.com/progate-git/mysite.git

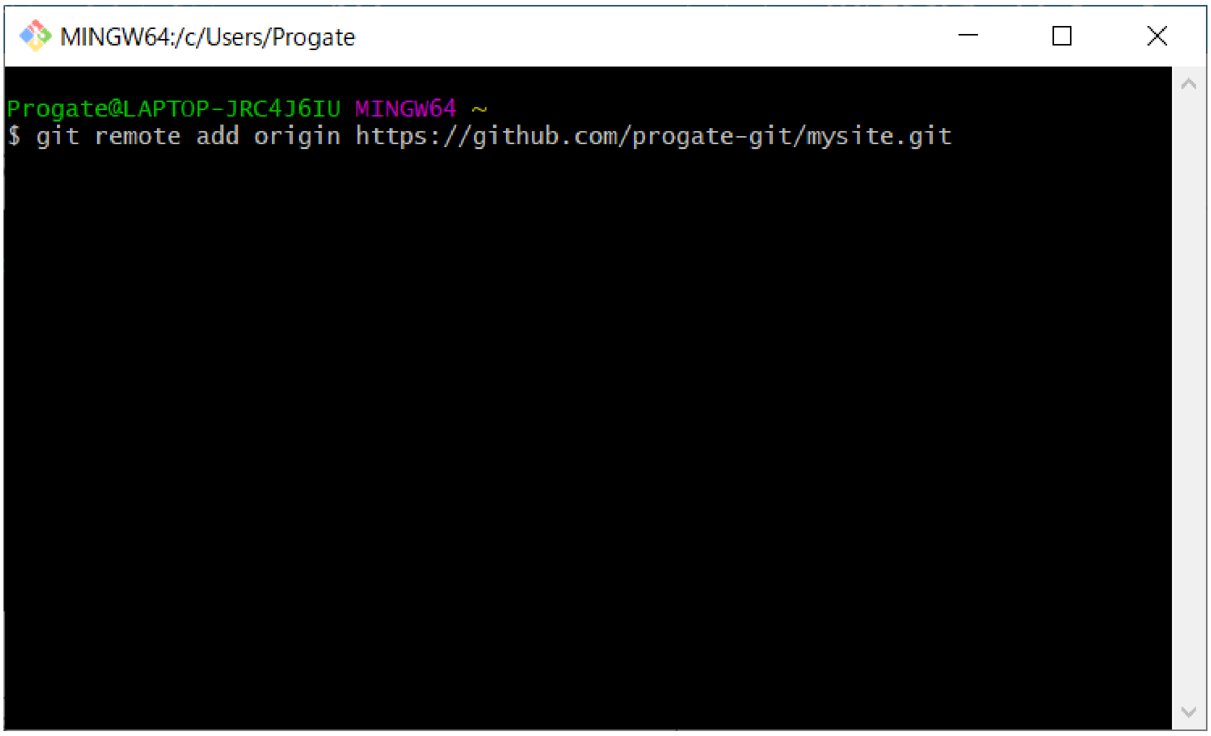
We recommend every repository include a README, LICENSE, and .gitignore.

Copy the URL

Once you have copied the line, replace the **<URL>** and run the following command:

```
git remote add origin <URL>
```

This command specifies the remote repository you want to transfer your local repository content from.



MINGW64:/c/Users/Progate

```
Progate@LAPTOP-JRC4J6IU MINGW64 ~
$ git remote add origin https://github.com/progate-git/mysite.git
```

Specify the remote repository

Push

Next, in order to push, you must create a file and commit it.

```
touch index.html
```

```
git add index.html
```

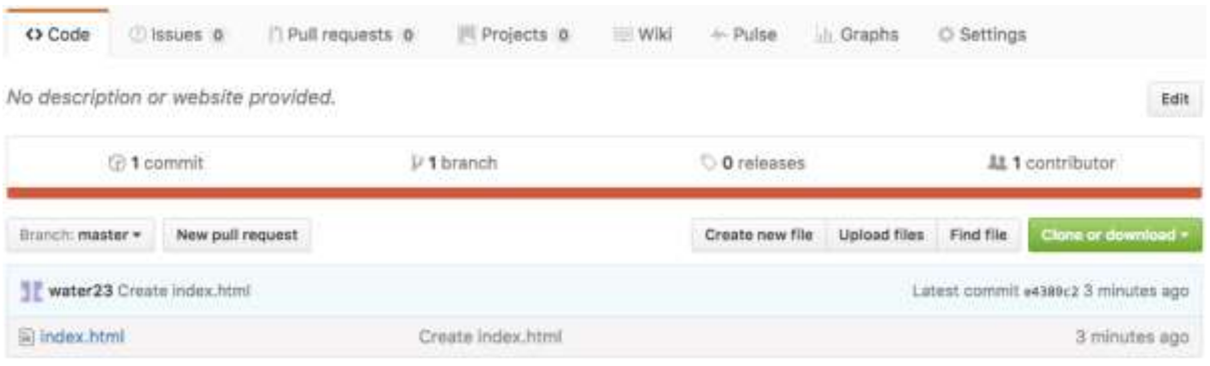
```
git commit -m "Create index.html"
```

Now you will use the "git push" command:

```
git push origin master
```

Let's check if the commit was pushed successfully on GitHub.

You should see an "index.html" file on GitHub if you have done everything correctly.



No description or website provided.

1 commit 1 branch 0 releases 1 contributor

water23 Create index.html Latest commit e4389c2 3 minutes ago

index.html Create index.html 3 minutes ago

Check that the commit was pushed

Check

Have you successfully pushed to remote repository?