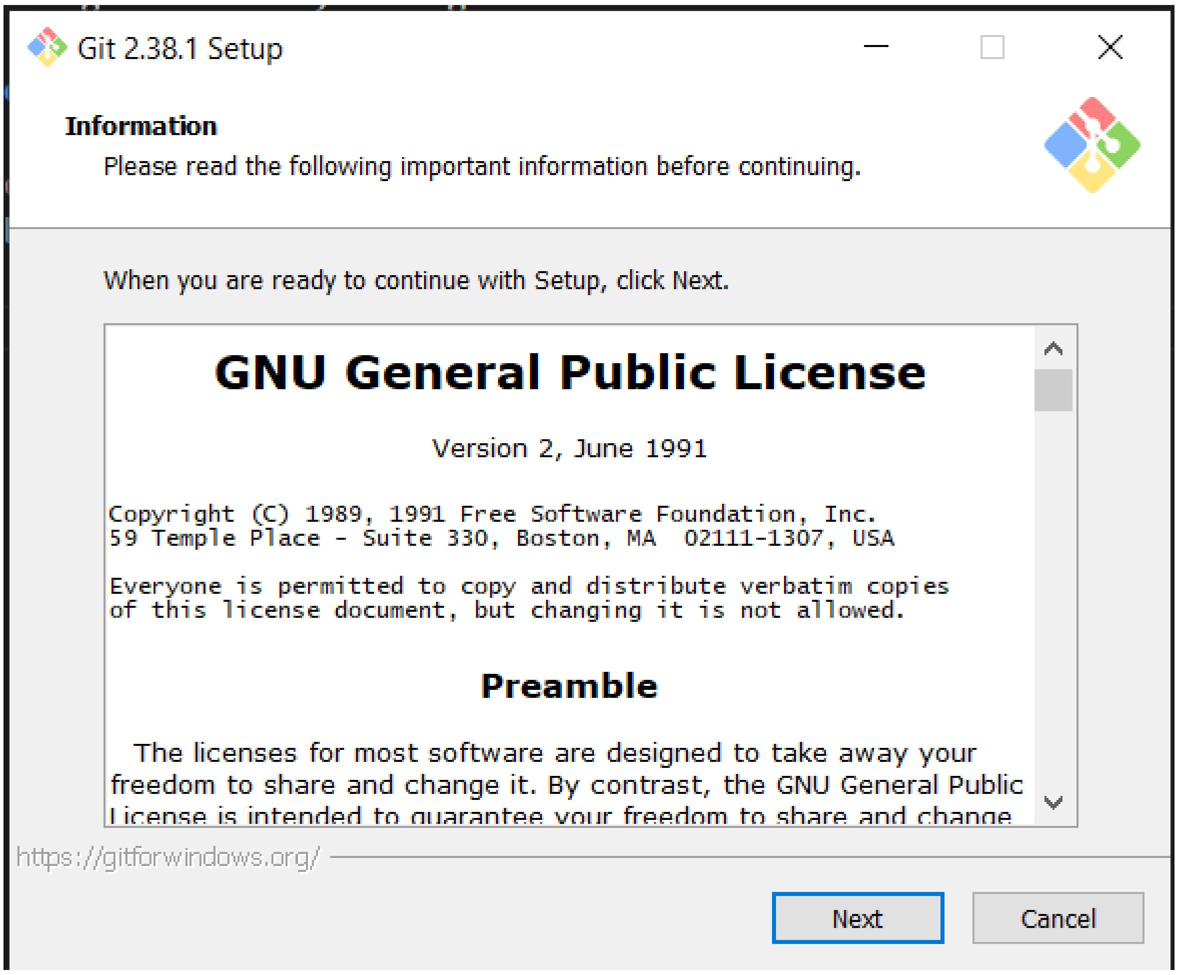
# Installing git for windows

This is a quick guide to installing git for Windows and a few troubleshooting tips are included at the end.

## Installing git for Windows

Download the latest Windows x64 installation from git scm Git Downloads

Once you have downloaded (and verified) start the setup. The following screens show the options as of version `2.38.1.windows.1`



**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

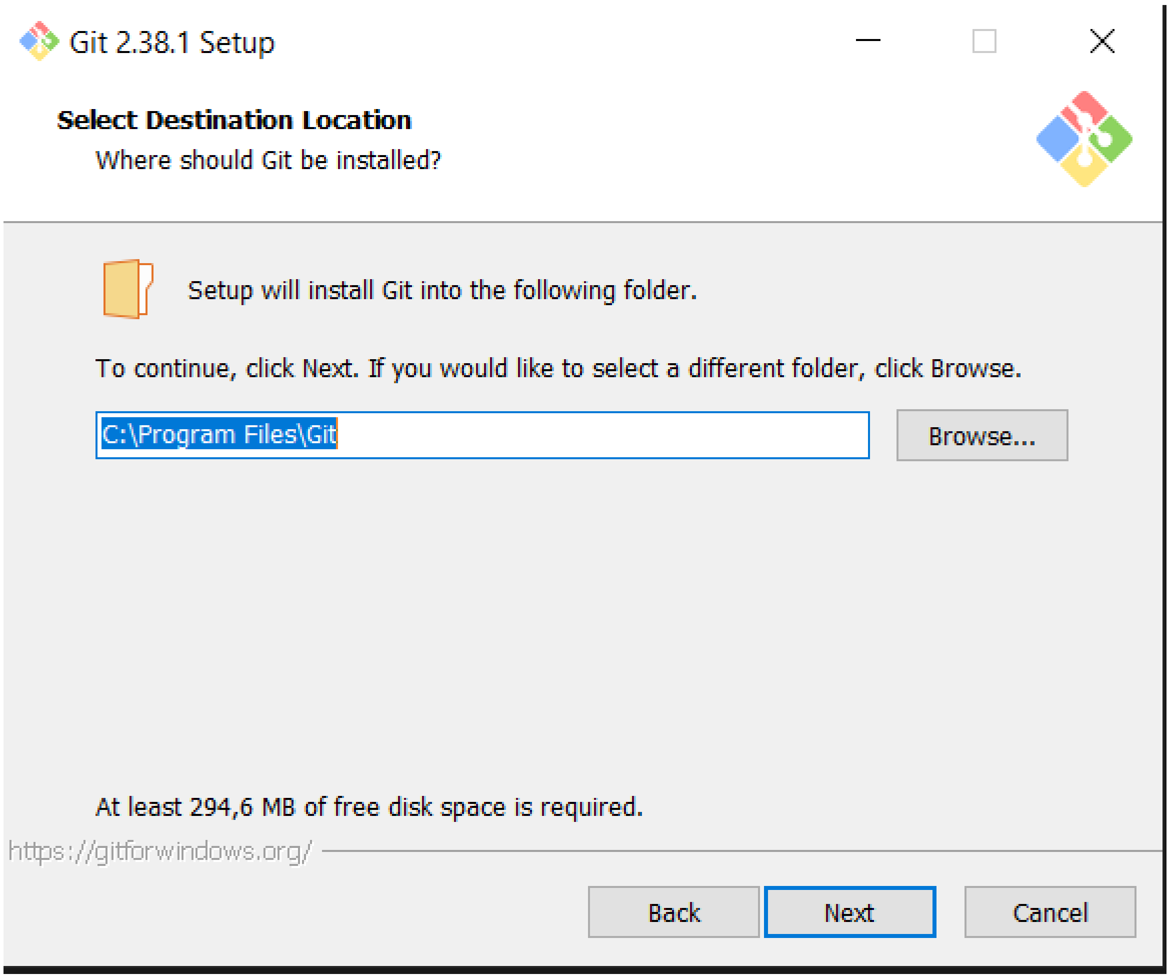
Copyright (C) 1989, 1991 Free Software Foundation, Inc.
59 Temple Place - Suite 330, Boston, MA 02111-1307, USA

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 to accept the license. The default installation location should be fine:


Git 2.38.1 Setup

Select Destination Location

Where should Git be installed?

Setup will install Git into the following folder.

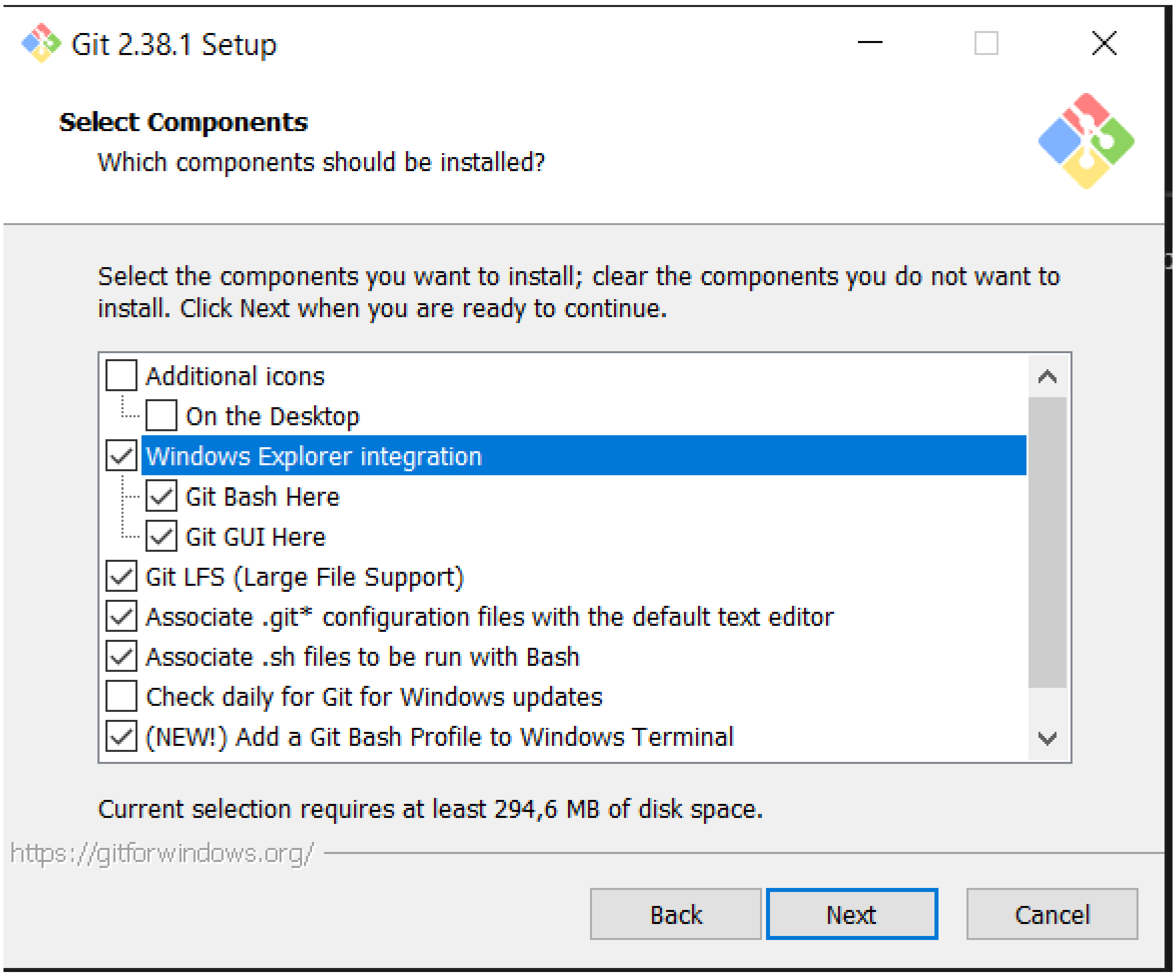
To continue, click Next. If you would like to select a different folder, click Browse.

C:\Program Files\Git

At least 294,6 MB of free disk space is required.

Click Next to confirm your installation location.

Use the screenshot below as a guideline:


Git 2.38.1 Setup

**Select Components**

Which components should be installed?

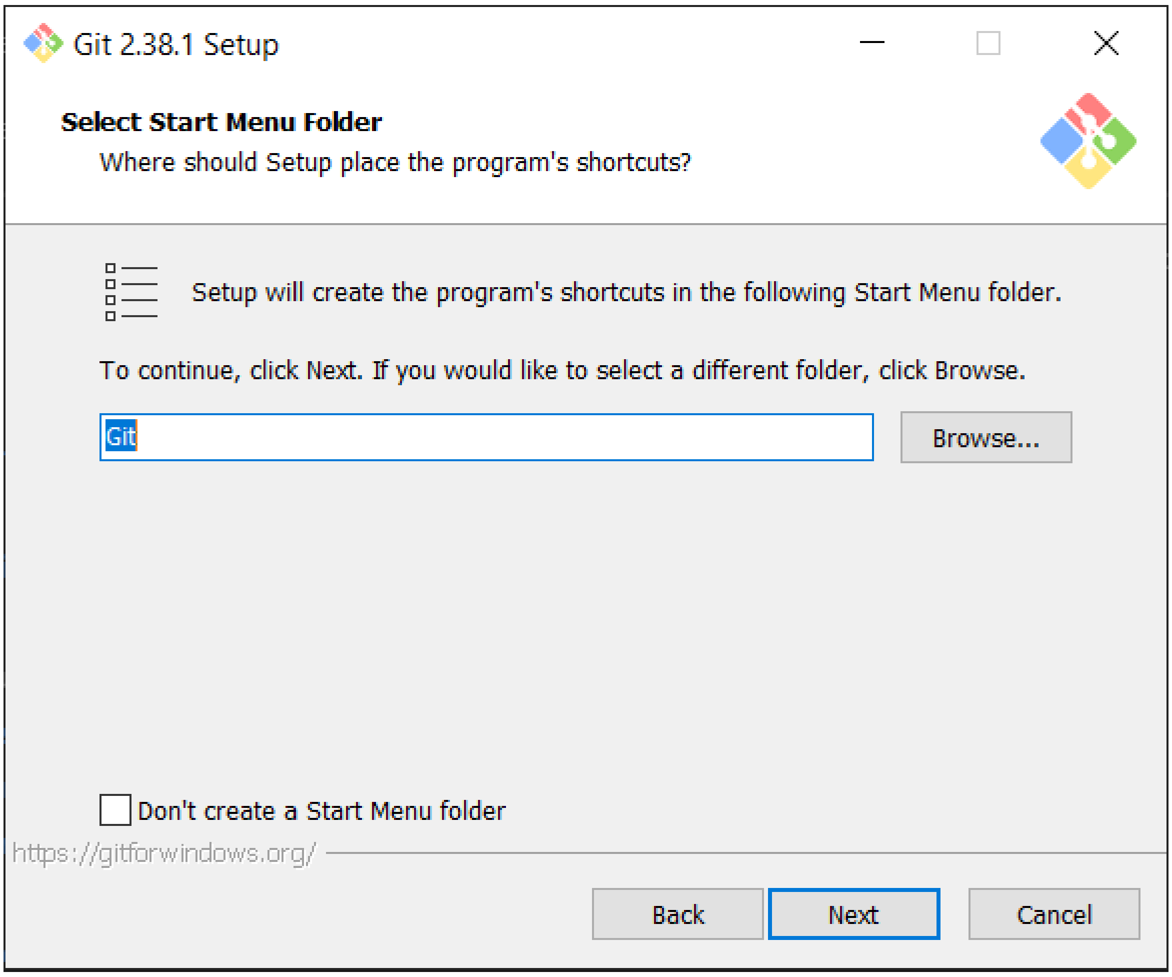
Select the components you want to install; clear the components you do not want to install. Click Next when you are ready to continue.

- [ ] 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
- [ ] Check daily for Git for Windows updates
- [x] (NEW!) Add a Git Bash Profile to Windows Terminal

Current selection requires at least 294,6 MB of disk space.

Click Next to confirm your selection.

Select your Start Menu Folder; the default should be fine.



**Select Start Menu Folder**

Where should Setup place the program's shortcuts?

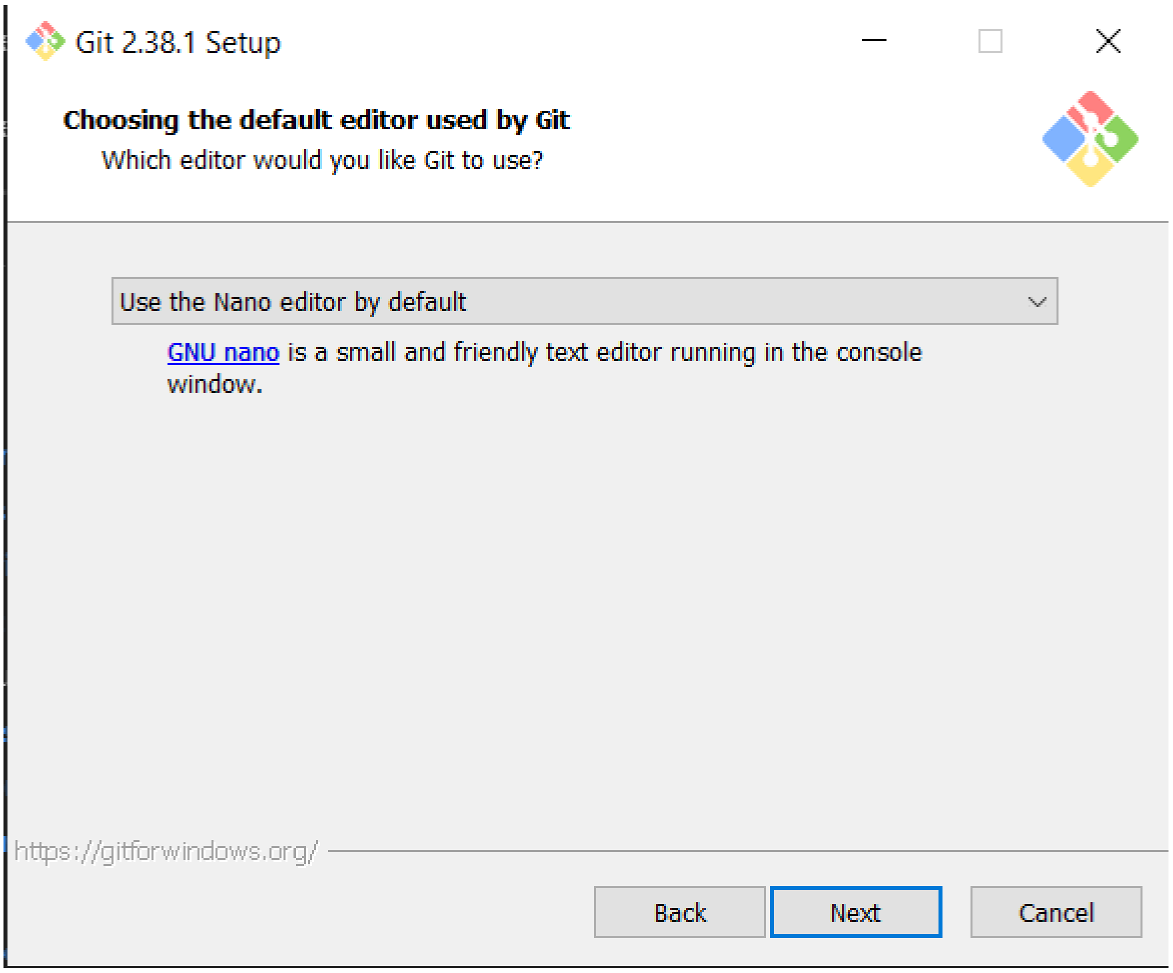
Setup will create the program's shortcuts in the following Start Menu folder.

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

Git

Click Next to confirm your selection.

It is recommended that you set the default editor to nano because the commands are a lot more intuitive if you are a Windows user. For example, `CTRL+S` will save the file.



**Choosing the default editor used by Git**

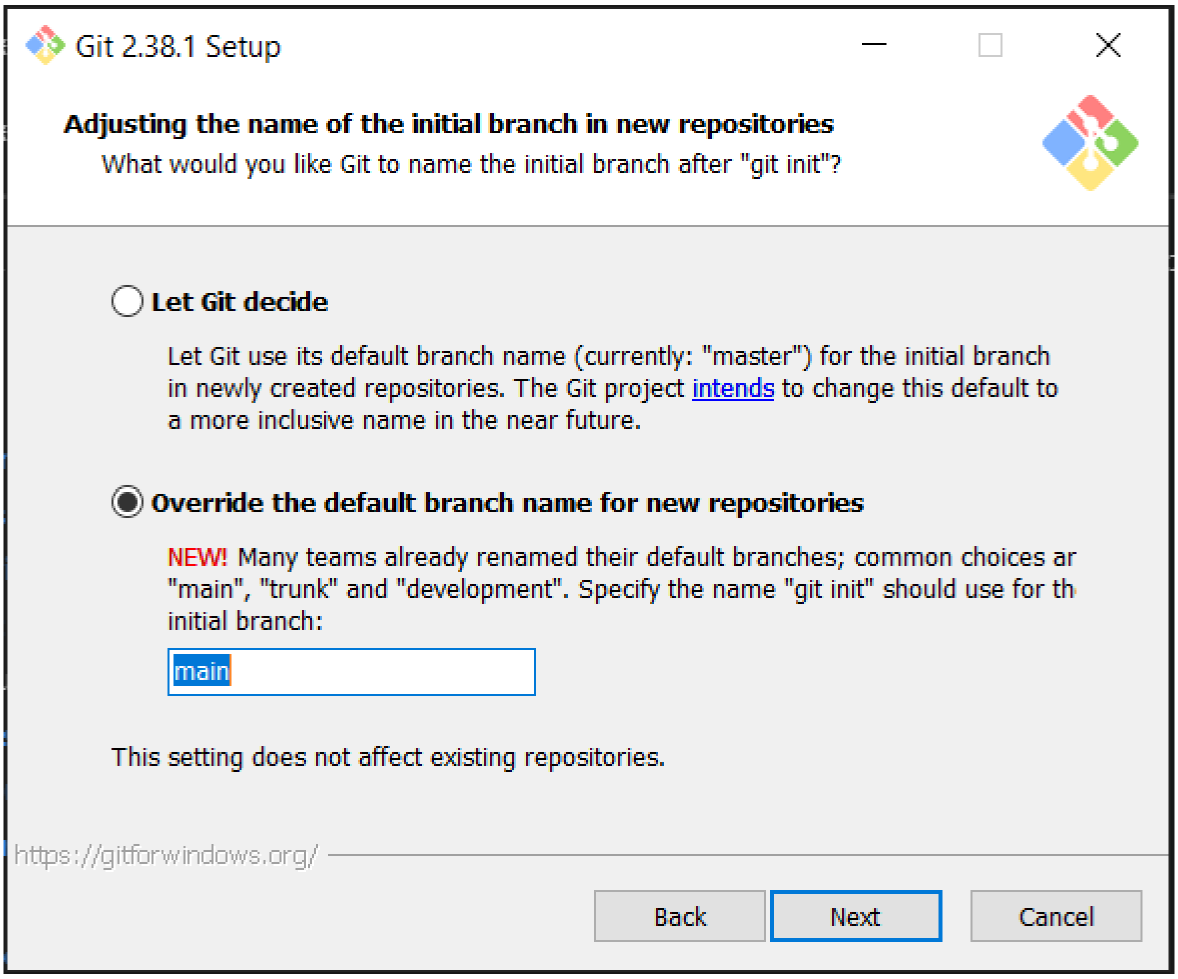
Which editor would you like Git to use?

Use the Nano editor by default

GNU nano is a small and friendly text editor running in the console window.

Click Next to confirm your selection.

It is recommended that you select `main` as your default branch, as this will align with GitHub's default branch naming convention.



**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**

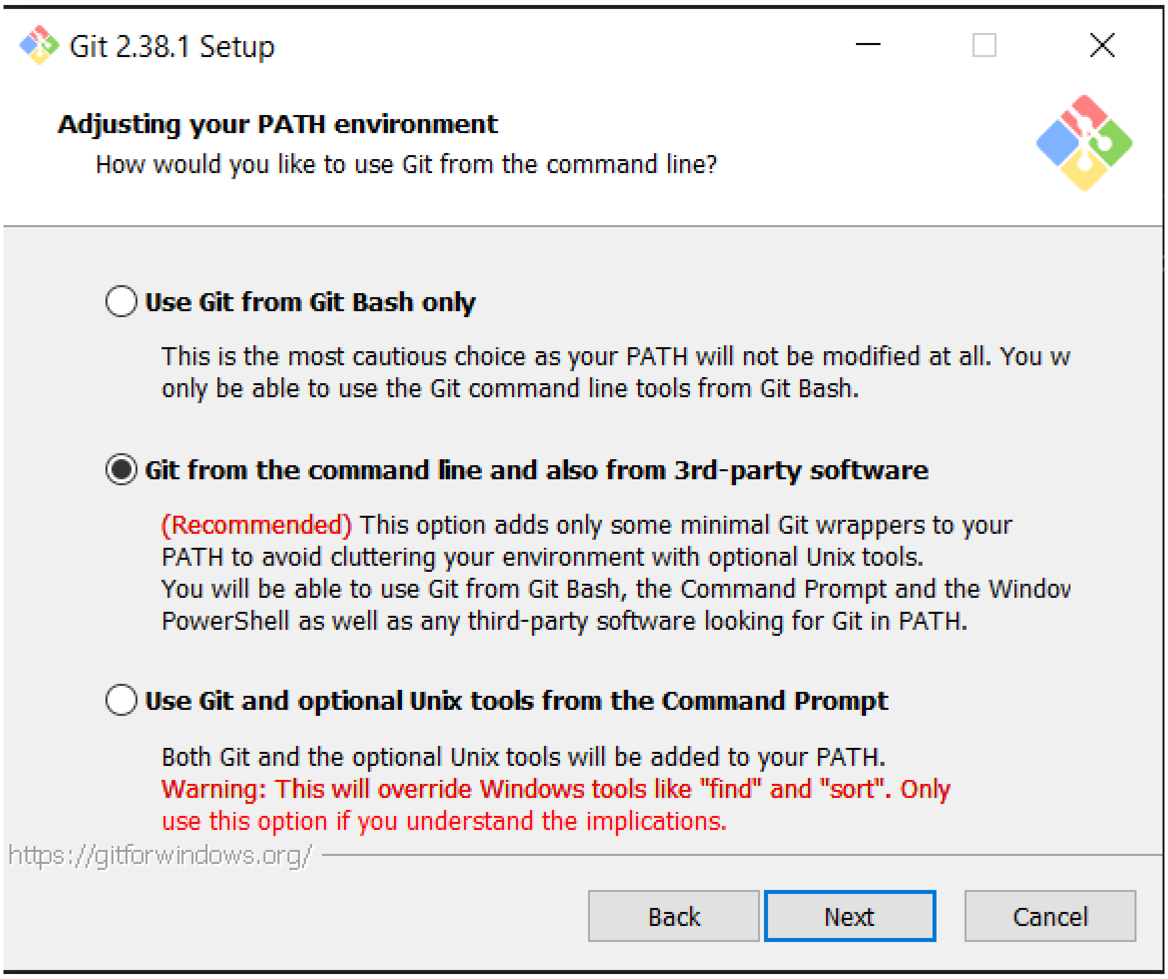
NEW! Many teams already renamed their default branches; common choices ar "main", "trunk" and "development". Specify the name "git init" should use for th initial branch:

main

This setting does not affect existing repositories.

Click Next to confirm your selection.

It is recommended that you leave this on the default setting `Git from the command line and also from 3rd-party software`


Git 2.38.1 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**

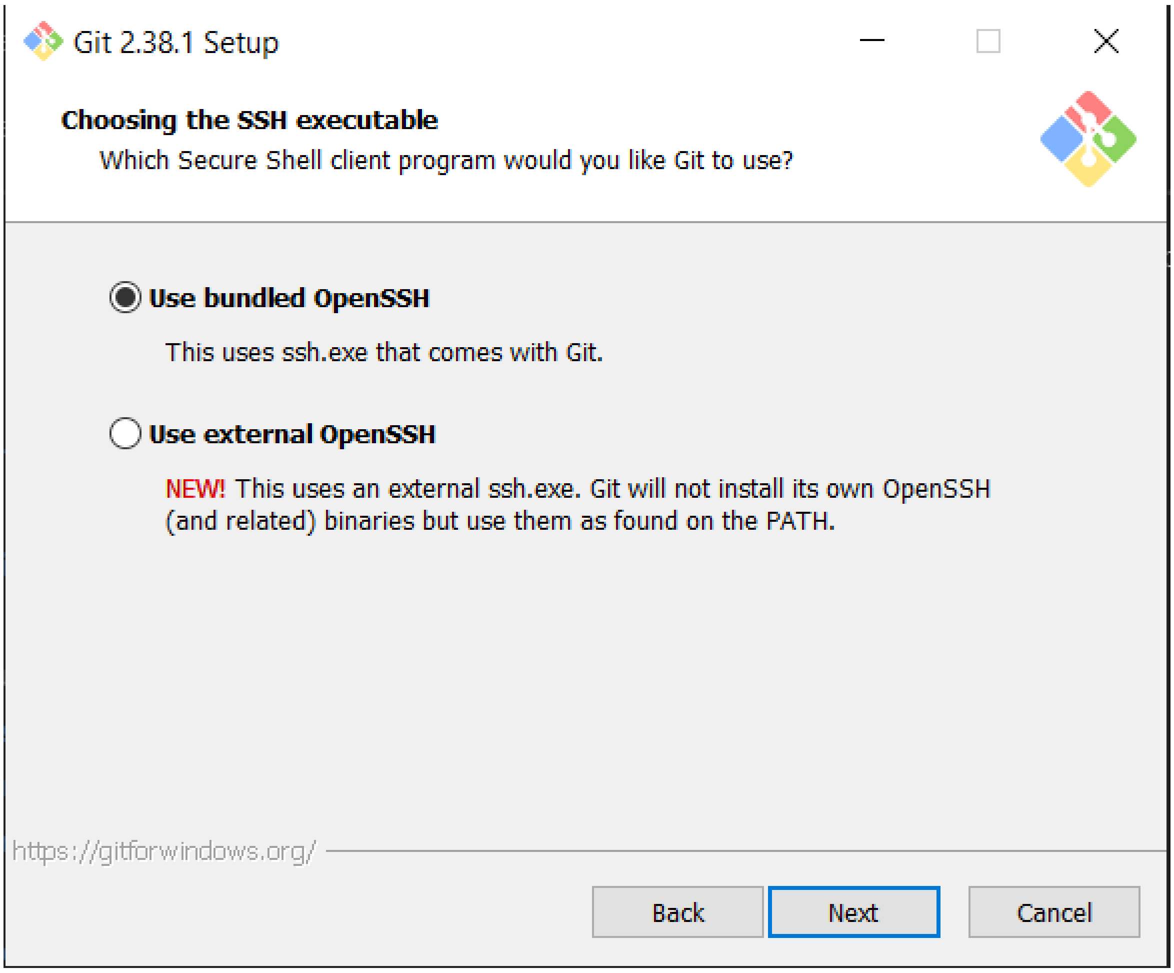
(Recommended) This option adds only some minimal Git wrappers to your
PATH to avoid cluttering your environment with optional Unix tools.
You will be able to use Git from Git Bash, the Command Prompt and the Windov
PowerShell as well as any third-party software looking for Git in PATH.

( ) **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.

Click Next to confirm your selection.

It is recommended to use the default: `Use bundled OpenSSH`. Unless you have already configured OpenSSH in your environment.


Git 2.38.1 Setup

**Choosing the SSH executable**

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

[✓ Use bundled OpenSSH]

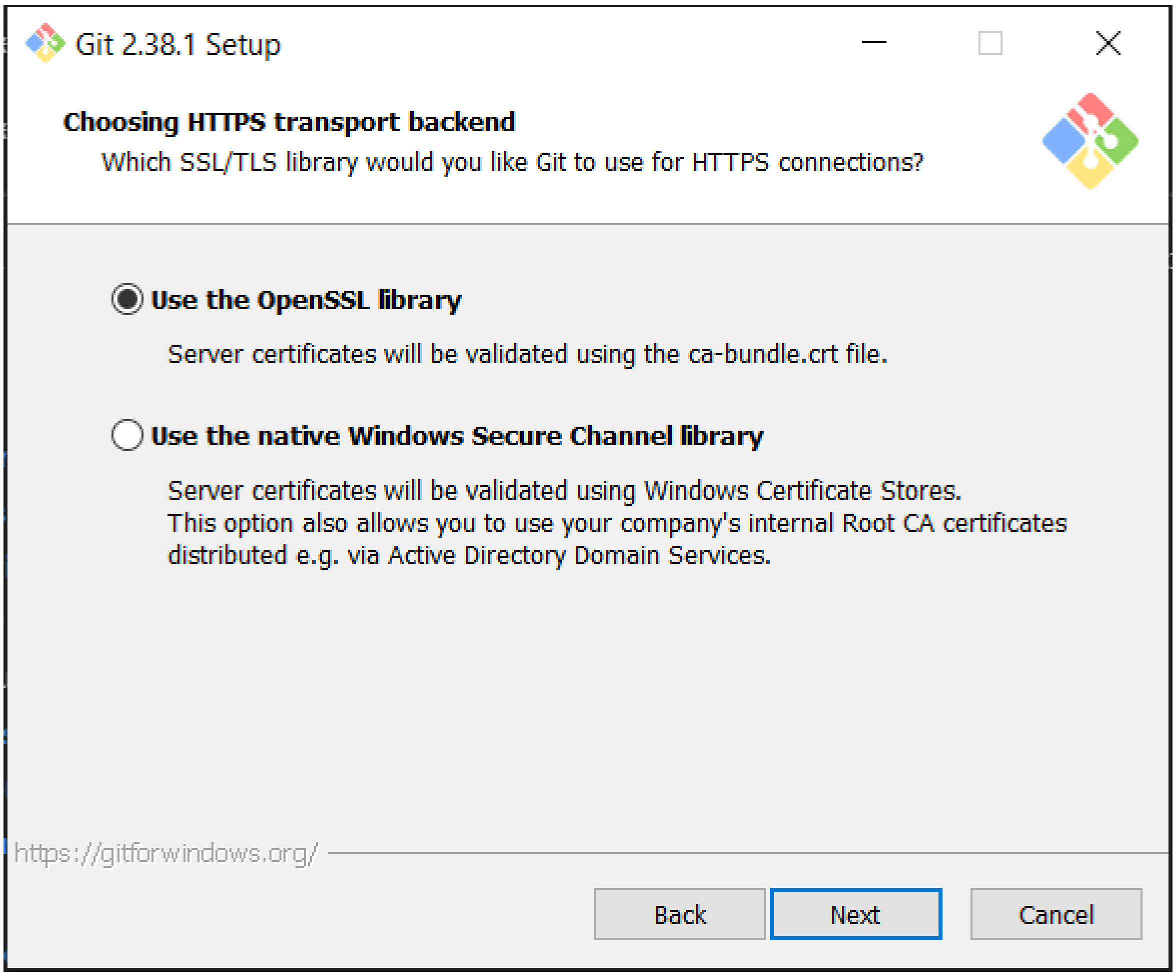
This uses ssh.exe that comes with Git.

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.

Click Next to confirm your selection.

It is recommended to use the default: `OpenSSL library`. Unless you have requirements to use internal CA certificates.


Git 2.38.1 Setup

**Choosing HTTPS transport backend**

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

(●) **Use the OpenSSL library**

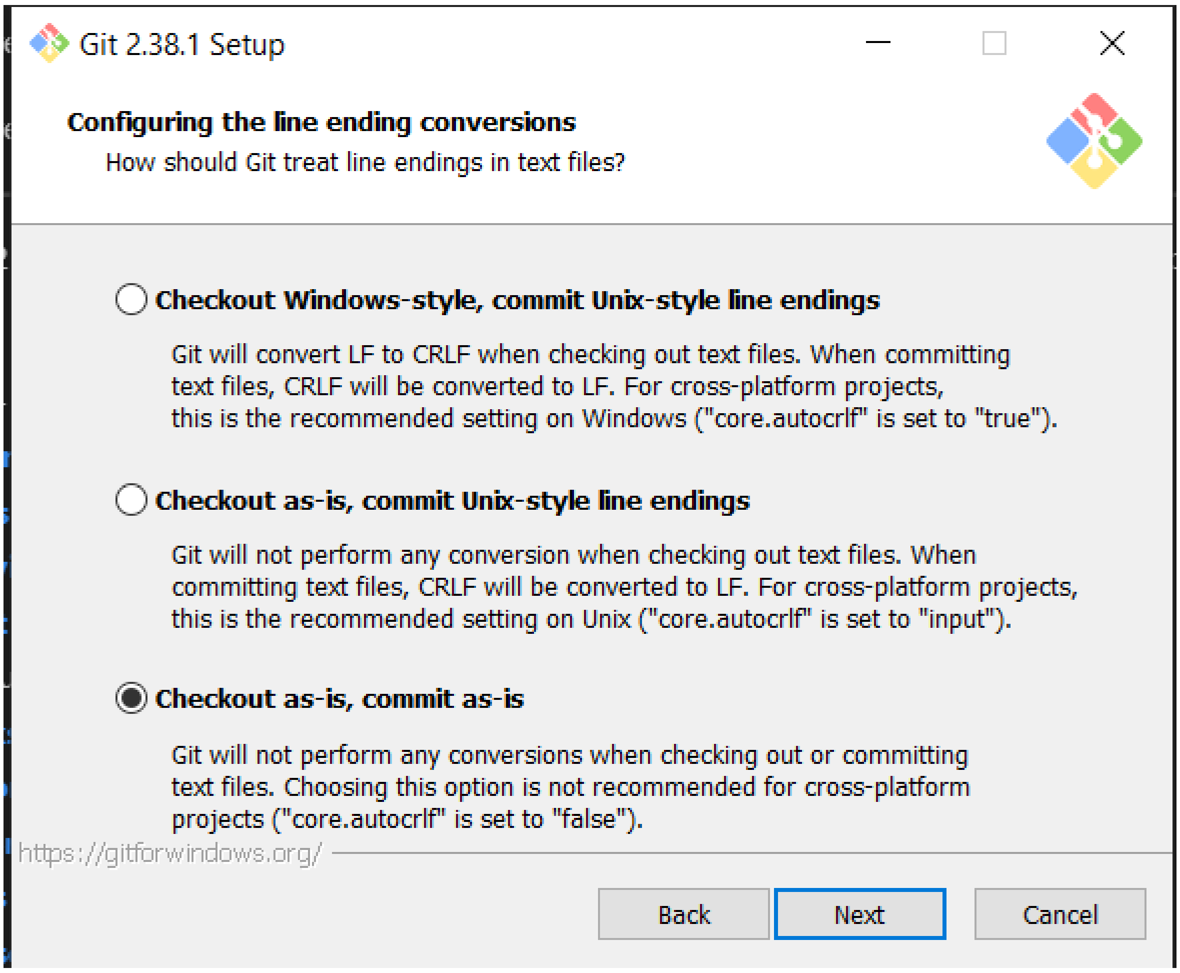
Server certificates will be validated using the ca-bundle.crt file.

( ) **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.

Click Next to confirm your selection.

It is recommended to use `Checkout as-is, commit as-is` as your default option. Discuss with your team if you prefer Unix-style line endings, in which case, use one of the first two options.


Git 2.38.1 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**

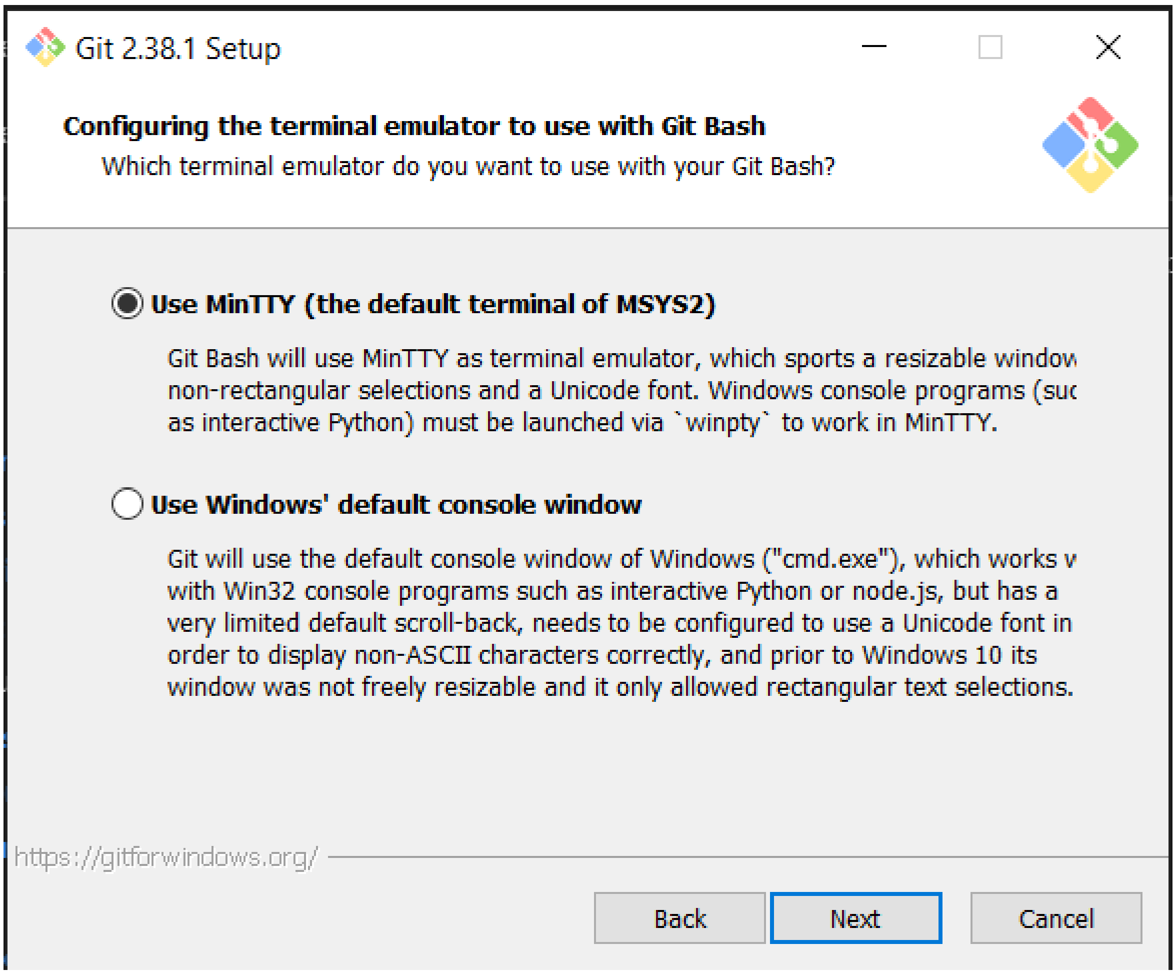
Git will not perform any conversion when checking out text files. When committing text files, CRLF will be converted to LF. For cross-platform projects, this is the recommended setting on Unix ("core.autocrlf" is set to "input").

[✓ **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").

Click Next to confirm your selection.

Using `MinTTY` is recommended:



Git 2.38.1 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)**]

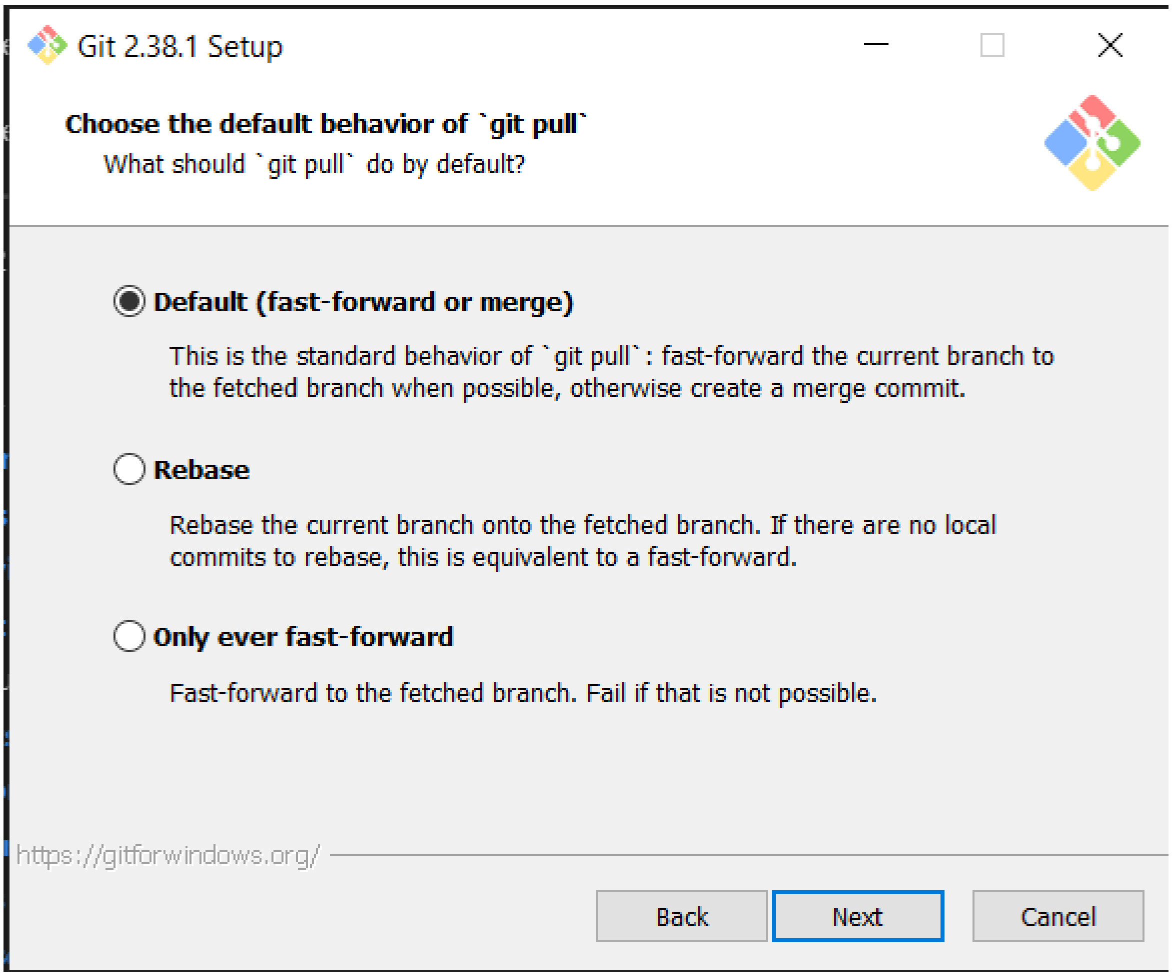
Git Bash will use MinTTY as terminal emulator, which sports a resizable window non-rectangular selections and a Unicode font. Windows console programs (suc as interactive Python) must be launched via `winpty` to work in MinTTY.

**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.

Click Next to confirm your selection.

It is recommended that you leave this on the default setting:



**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

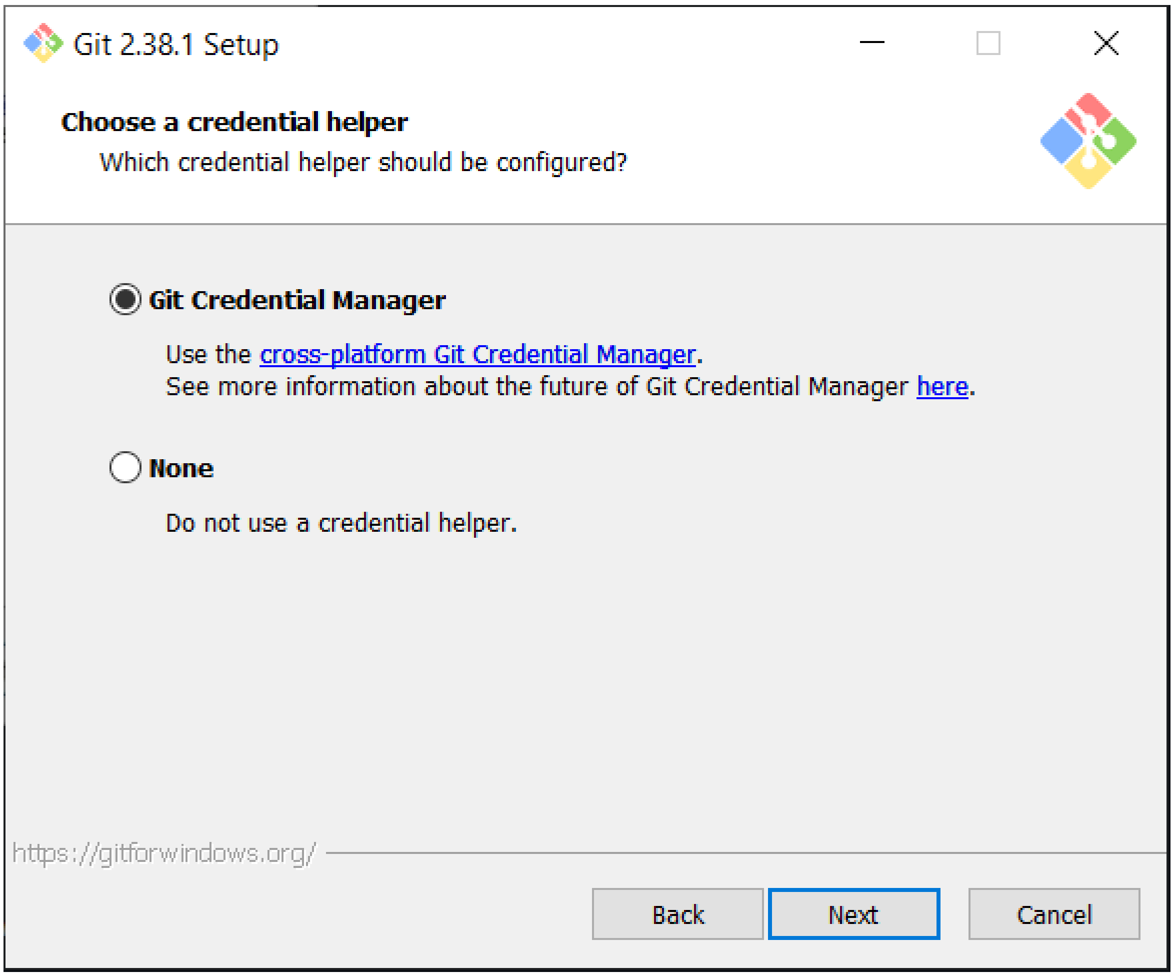
Rebase the current branch onto the fetched branch. If there are no local commits to rebase, this is equivalent to a fast-forward.

Only ever fast-forward

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

Click Next to confirm your selection.

It is recommended that you leave this on the default setting:



**Choose a credential helper**

Which credential helper should be configured?

[● **Git Credential Manager**]

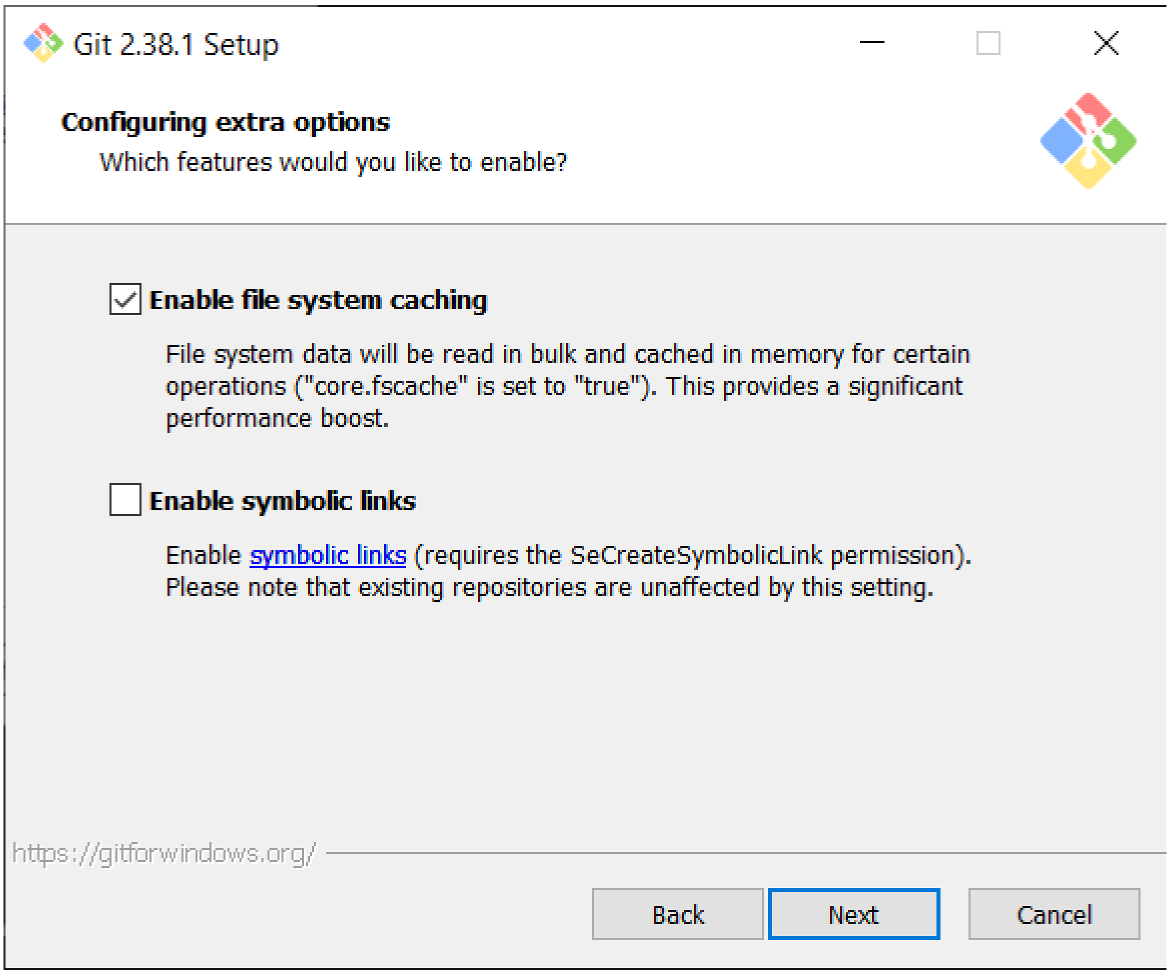
Use the cross-platform Git Credential Manager.
See more information about the future of Git Credential Manager here.

○ **None**

Do not use a credential helper.

Click Next to confirm your selection.

It is recommended that you leave this on the default setting:



Git 2.38.1 Setup

**Configuring extra options**

Which features would you like to enable?

[✓] **Enable file system caching**

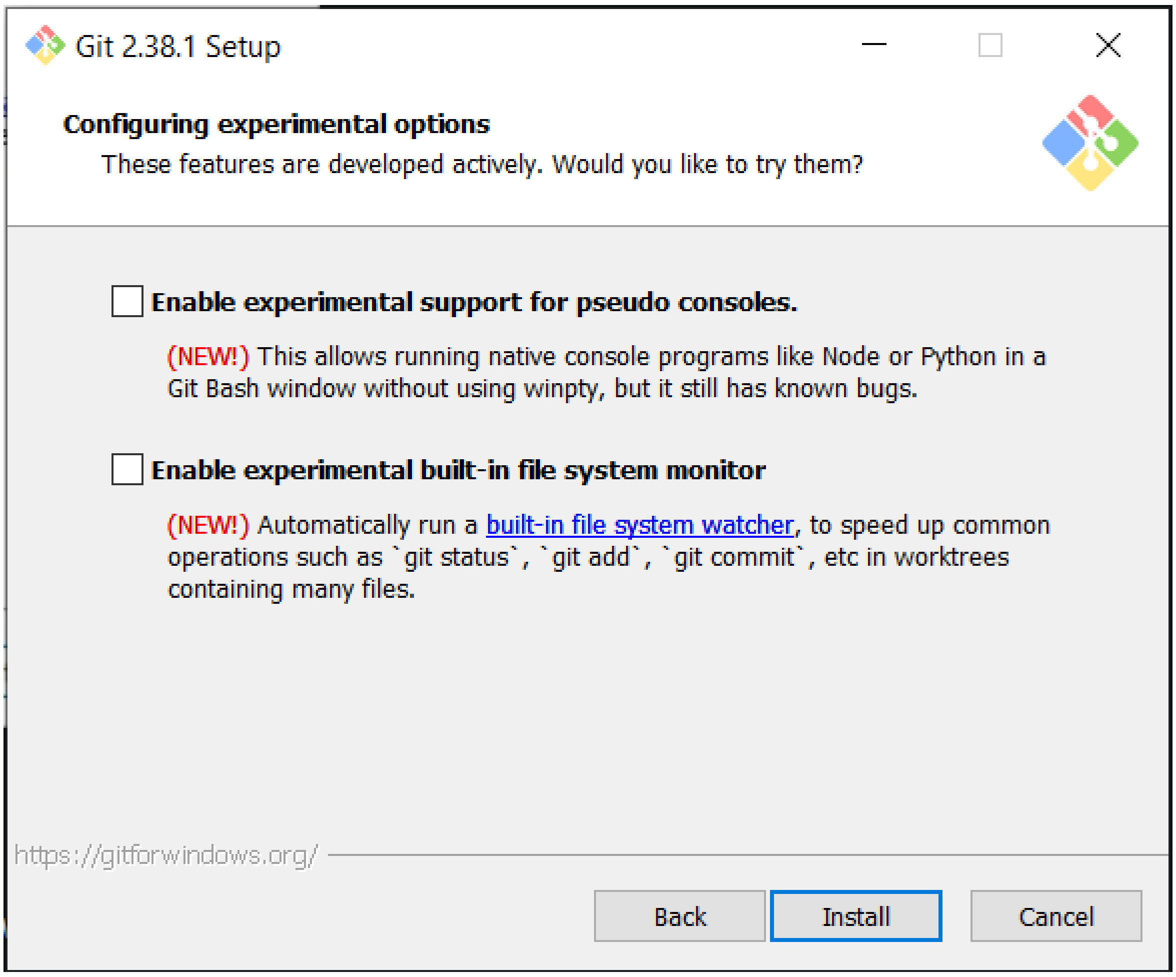
File system data will be read in bulk and cached in memory for certain operations ("core.fscache" is set to "true"). This provides a significant performance boost.

[ ] **Enable symbolic links**

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

Click Next to confirm your selection.

It is recommended no experimental features are installed.


Git 2.38.1 Setup

**Configuring experimental options**

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

☐ **Enable experimental support for pseudo consoles.**

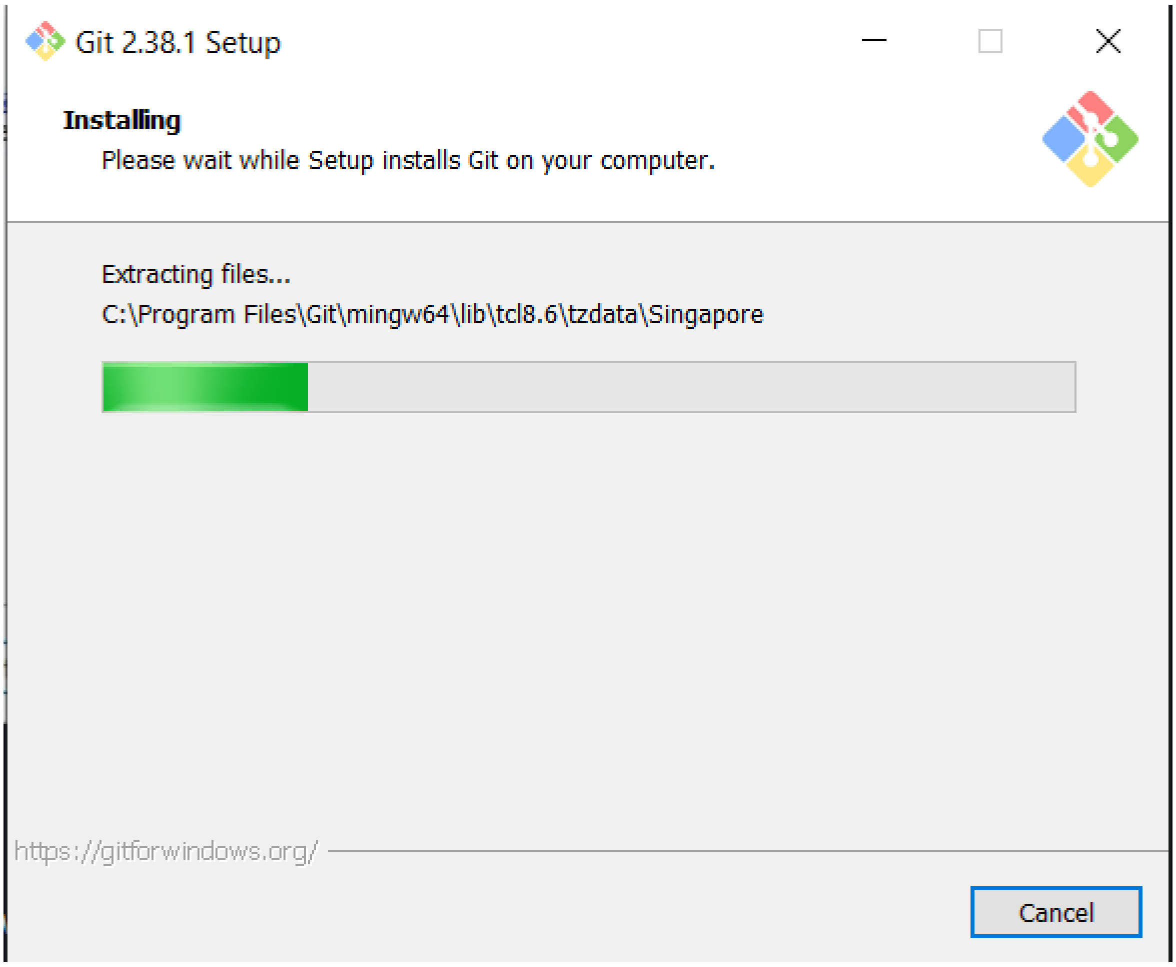
(NEW!) This allows running native console programs like Node or Python in a Git Bash window without using winpty, but it still has known bugs.

☐ **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/

Click Next to start installing.


Git 2.38.1 Setup

**Installing**

Please wait while Setup installs Git on your computer.

Extracting files...
C:\Program Files\Git\mingw64\lib\tcl8.6\tzdata\Singapore

https://gitforwindows.org/

At the end of the installation, it is recommended that you launch Git Bash to start the configuration of git in Windows.

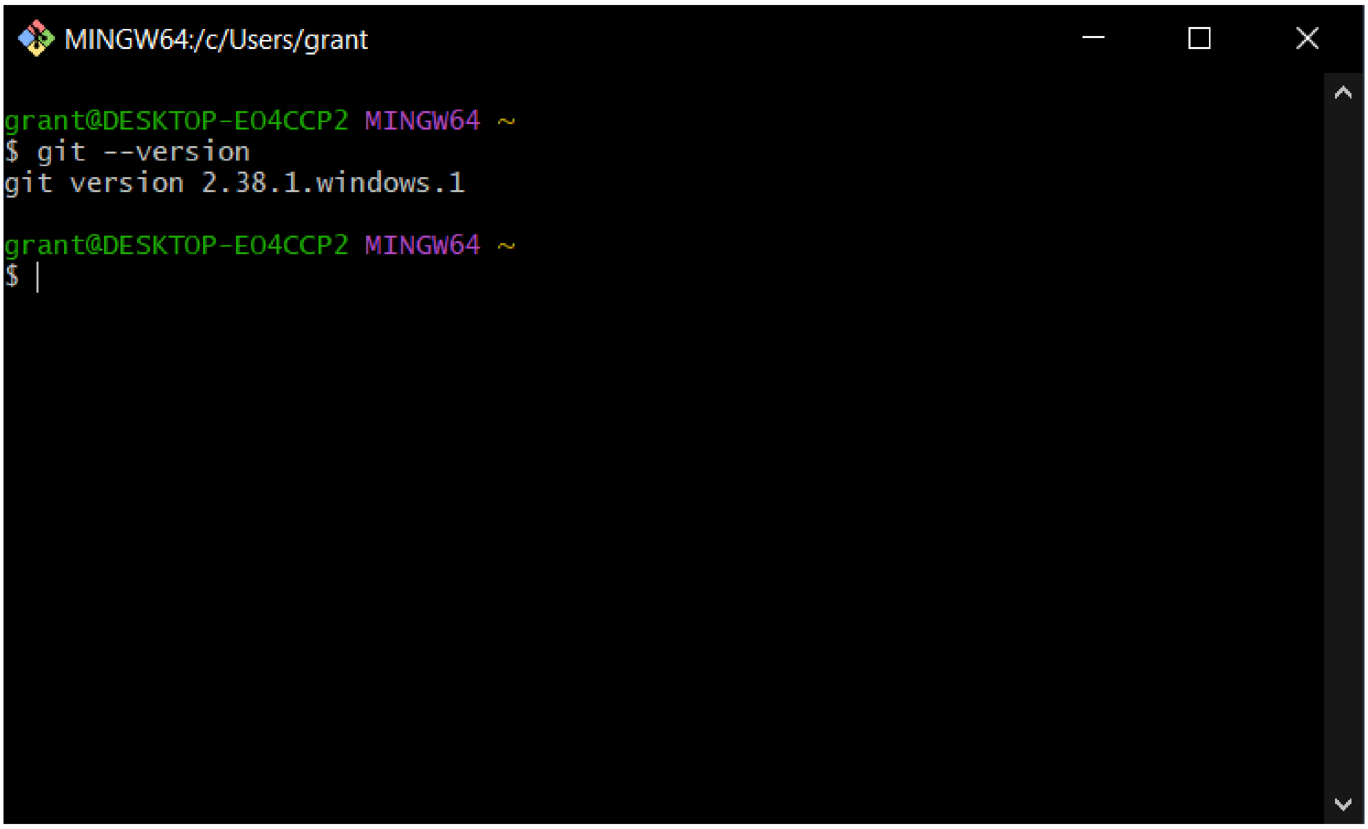
## Configuring git in Windows

If you did not already open `Git Bash` at the end of the installation above, then open `Git Bash` from your Start menu.

Check if git is installed by running

```
git --version
```

For example:



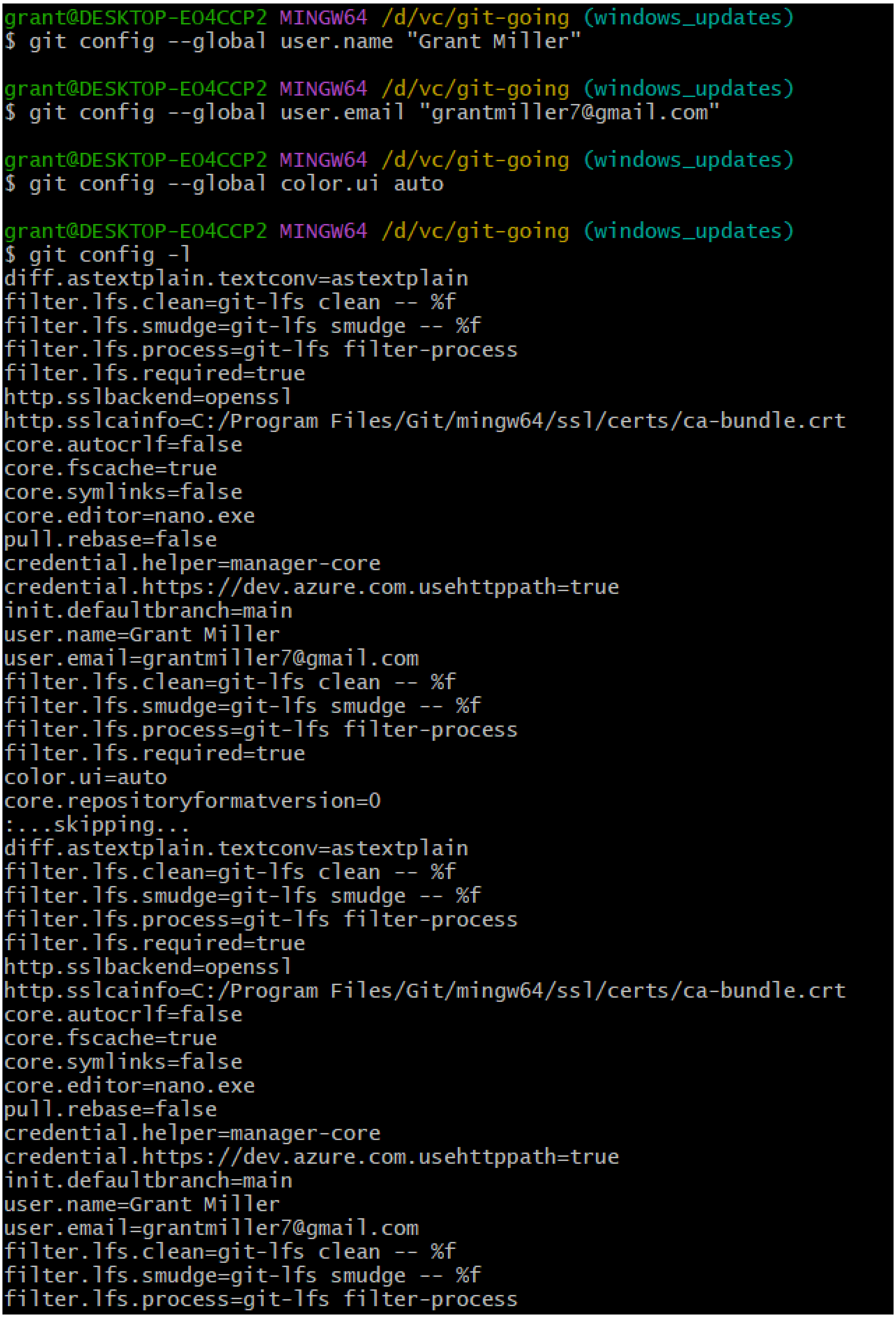
```
grant@DESKTOP-EO4CCP2 MINGW64 ~
$ git --version
git version 2.38.1.windows.1

grant@DESKTOP-EO4CCP2 MINGW64 ~
$ |
```

Now you need to configure git in your environment. Remember to replace the `"User Name"` and `"email@domain.co.za"` with your own name and email address:

```
git config --global user.name "User Name"
git config --global user.email "email@domain.co.za"
git config --global color.ui auto
git config -l
```

For example:



```
grant@DESKTOP-EO4CCP2 MINGW64 /d/vc/git-going (windows_updates)
$ git config --global user.name "Grant Miller"

grant@DESKTOP-EO4CCP2 MINGW64 /d/vc/git-going (windows_updates)
$ git config --global user.email "grantmiller7@gmail.com"

grant@DESKTOP-EO4CCP2 MINGW64 /d/vc/git-going (windows_updates)
$ git config --global color.ui auto

grant@DESKTOP-EO4CCP2 MINGW64 /d/vc/git-going (windows_updates)
$ git config -l
diff.astextplain.textconv=astextplain
filter.lfs.clean=git-lfs clean -- %f
filter.lfs.smudge=git-lfs smudge -- %f
filter.lfs.process=git-lfs filter-process
filter.lfs.required=true
http.sslbackend=openssl
http.sslcainfo=C:/Program Files/Git/mingw64/ssl/certs/ca-bundle.crt
core.autocrlf=false
core.fscache=true
core.symlinks=false
core.editor=nano.exe
pull.rebase=false
credential.helper=manager-core
credential.https://dev.azure.com.usehttppath=true
init.defaultbranch=main
user.name=Grant Miller
user.email=grantmiller7@gmail.com
filter.lfs.clean=git-lfs clean -- %f
filter.lfs.smudge=git-lfs smudge -- %f
filter.lfs.process=git-lfs filter-process
filter.lfs.required=true
color.ui=auto
core.repositoryformatversion=0
:...skipping...
diff.astextplain.textconv=astextplain
filter.lfs.clean=git-lfs clean -- %f
filter.lfs.smudge=git-lfs smudge -- %f
filter.lfs.process=git-lfs filter-process
filter.lfs.required=true
http.sslbackend=openssl
http.sslcainfo=C:/Program Files/Git/mingw64/ssl/certs/ca-bundle.crt
core.autocrlf=false
core.fscache=true
core.symlinks=false
core.editor=nano.exe
pull.rebase=false
credential.helper=manager-core
credential.https://dev.azure.com.usehttppath=true
init.defaultbranch=main
user.name=Grant Miller
user.email=grantmiller7@gmail.com
filter.lfs.clean=git-lfs clean -- %f
filter.lfs.smudge=git-lfs smudge -- %f
filter.lfs.process=git-lfs filter-process
```

Generate your SSH key

It is recommended to use SSH over HTTPS as we have found it to be more stable. This section explains how to create and configure your SSH Key so that you can use SSH with GitHub.

If you do not already have `Git Bash` open, then open `Git Bash` from your Start menu.

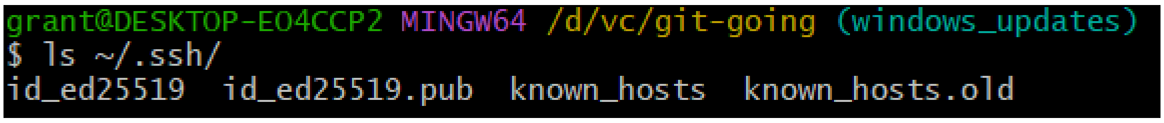
Generate your SSH key by using the following command. Remember to replace `"your_email@example.com"` with your own email address. *DO NOT ADD A PASSPHRASE*. If you add a pass phrase you will need to enter it every time you use SSH commands in git, which will be most of the time. This will slow you down a lot. Accept the default filename and location by pressing Enter on your keyboard.

```
ssh-keygen -t ed25519 -C "your_email@example.com"
```

The above command should create two files in your home SSH diretory called `id_ed25519` and `id_ed25519.pub`. This is a private and public key pair. Check that you have the following files by running this command:

```
ls ~/.ssh/
```

Example output:

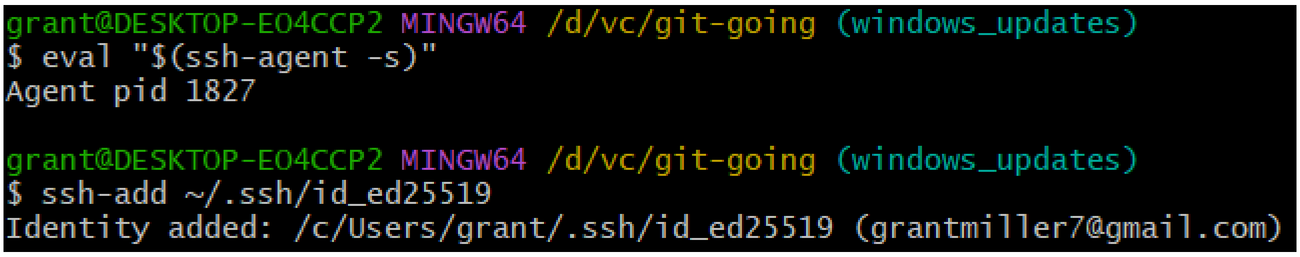


```
grant@DESKTOP-E04CCP2 MINGW64 /d/vc/git-going (windows_updates)
$ ls ~/.ssh/
id_ed25519  id_ed25519.pub  known_hosts  known_hosts.old
```

Next you need to add this key to your SSH agent. Run the following commands:

```
eval "$(ssh-agent -s)"
ssh-add ~/.ssh/id_ed25519
```

Example output:



```
grant@DESKTOP-E04CCP2 MINGW64 /d/vc/git-going (windows_updates)
$ eval "$(ssh-agent -s)"
Agent pid 1827

grant@DESKTOP-E04CCP2 MINGW64 /d/vc/git-going (windows_updates)
$ ssh-add ~/.ssh/id_ed25519
Identity added: /c/Users/grant/.ssh/id_ed25519 (grantmiller7@gmail.com)
```

Useful reference

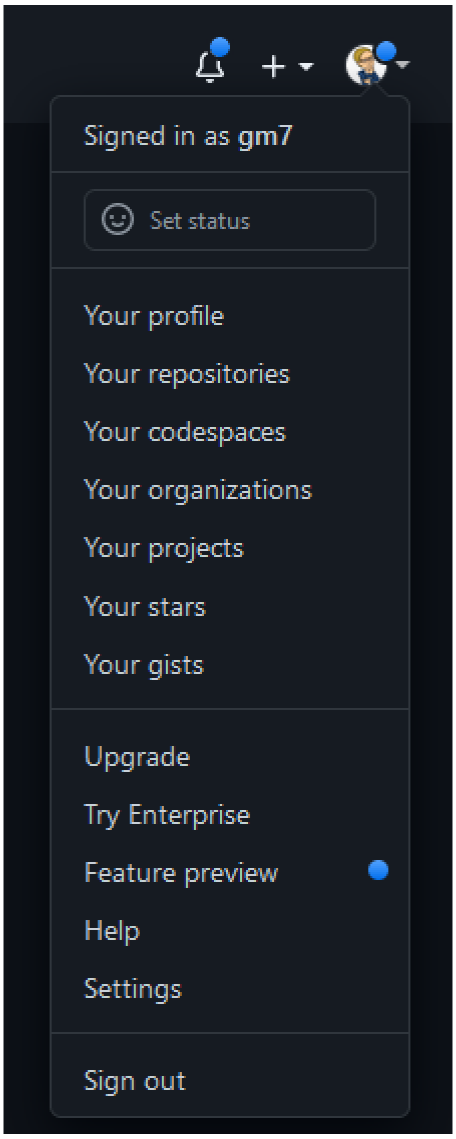
### Adding your SSH key to GitHub

Once you have created your SSH key and registered it with the SSH agent, you can add the public part of the key to GitHub.

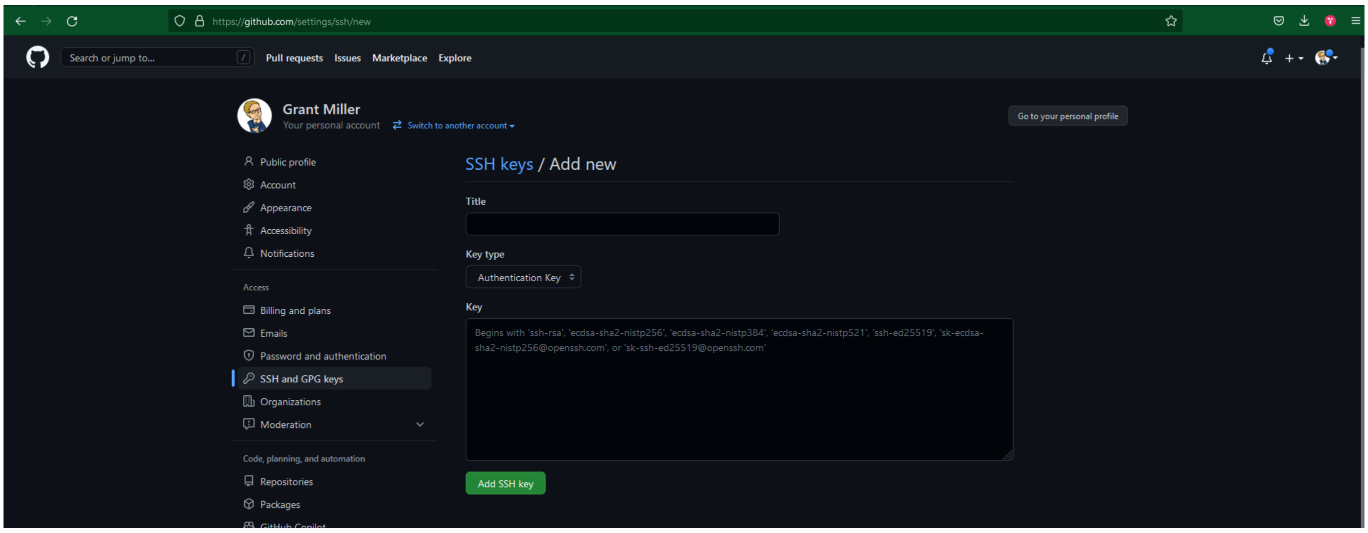
Remember to keep your private key safe and never to share it. Only the public key can be shared.

[Login to Github.com]

Once you have logged in, click on the user icon on the top right of the screen and select `Settings`



Select `SSH and GPG Keys` on the left, then select `New SSH Key` which is a green button on the right of your screen.



SSH keys / Add new

Title

Key type: Authentication Key

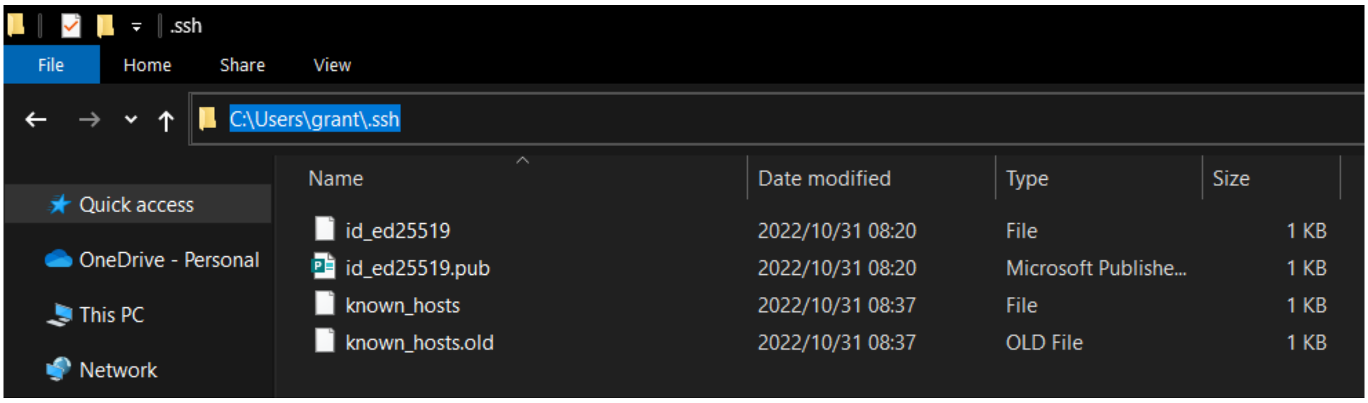
Key

Begins with 'ssh-rsa', 'ecdsa-sha2-nistp256', 'ecdsa-sha2-nistp384', 'ecdsa-sha2-nistp521', 'ssh-ed25519', 'sk-ecdsa-sha2-nistp256@openssh.com', or 'sk-ssh-ed25519@openssh.com'

Enter a meaningful title for your key. Typically you use the name of your machine, or the organisation that this key is connected with.

The body of the key needs to be the content of your public key file. Follow these steps to fill in the content of your public key on GitHub.com:

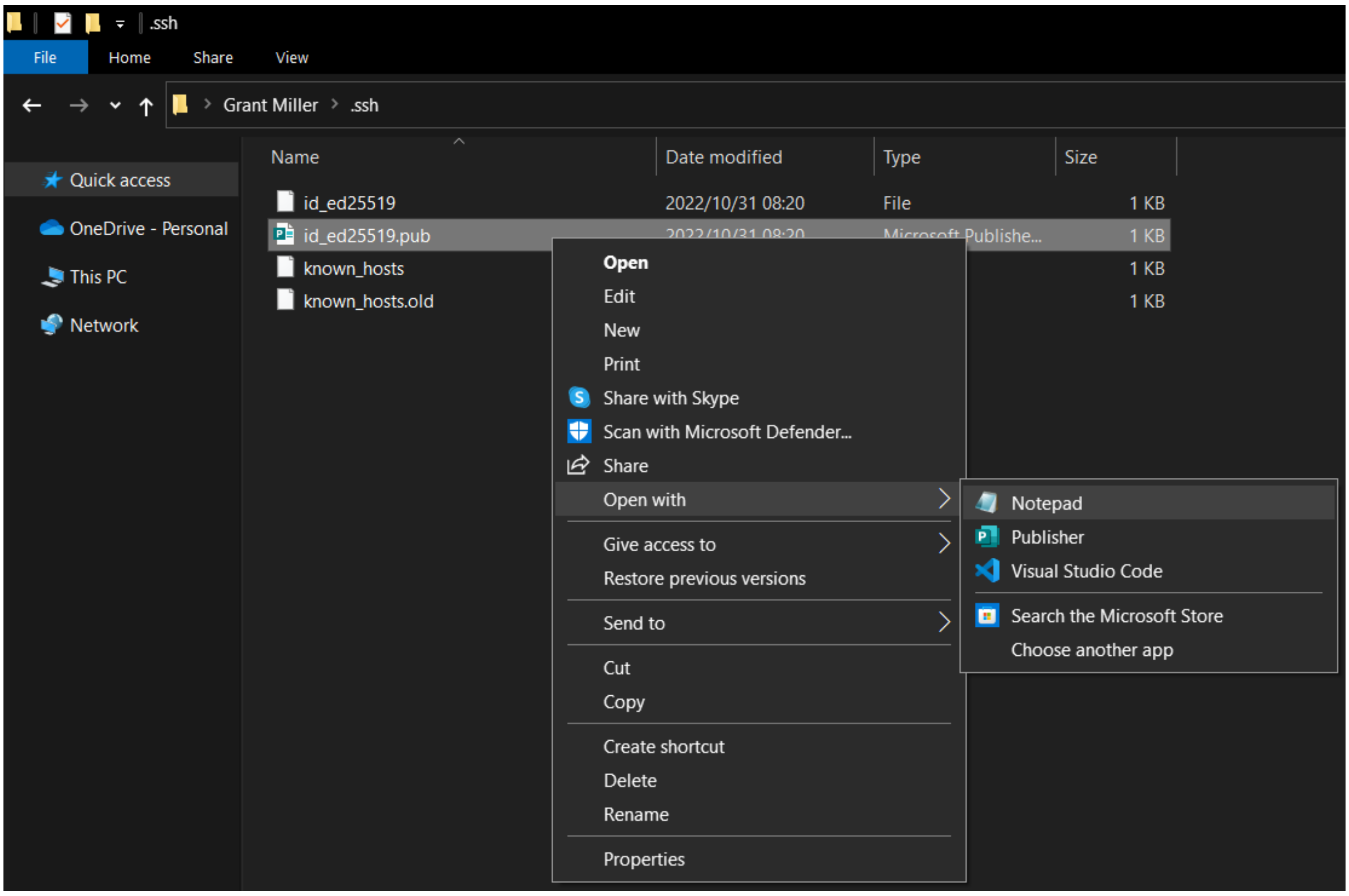
- Open Windows Explorer (WIN+E)
- Navigate to your home directory for SSH. For example: `c:\Users\<your windows username>\.ssh`



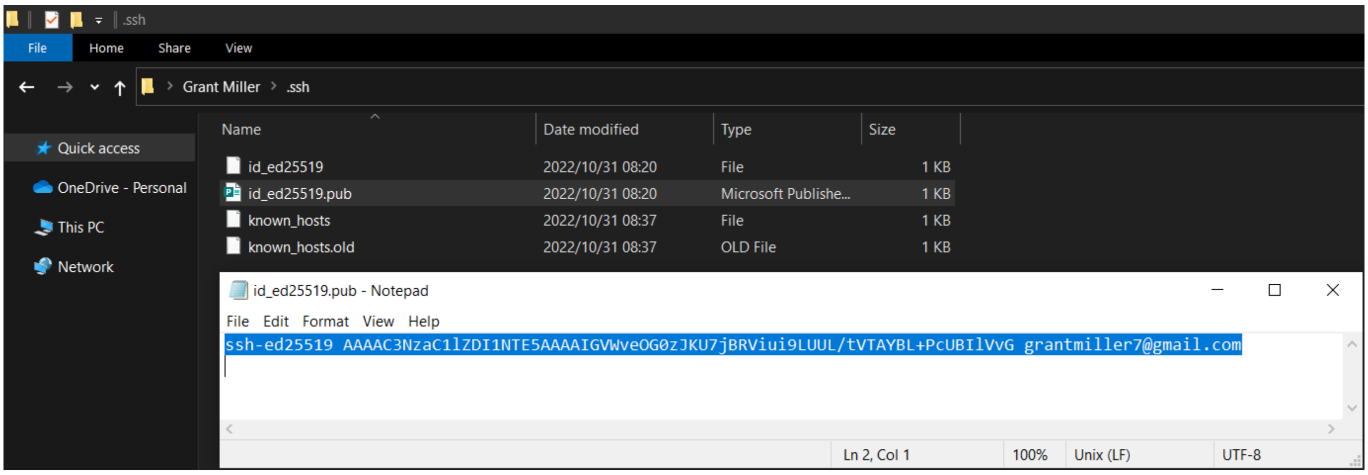
C:\Users\grant\.ssh

| Name | Date modified | Type | Size |
|---|---|---|---|
| id_ed25519 | 2022/10/31 08:20 | File | 1 KB |
| id_ed25519.pub | 2022/10/31 08:20 | Microsoft Publishe... | 1 KB |
| known_hosts | 2022/10/31 08:37 | File | 1 KB |
| known_hosts.old | 2022/10/31 08:37 | OLD File | 1 KB |

- Right click your public key file and open with Notepad. For example:



- Copy all of the content (CTRL+A, CTRL+C). For example:



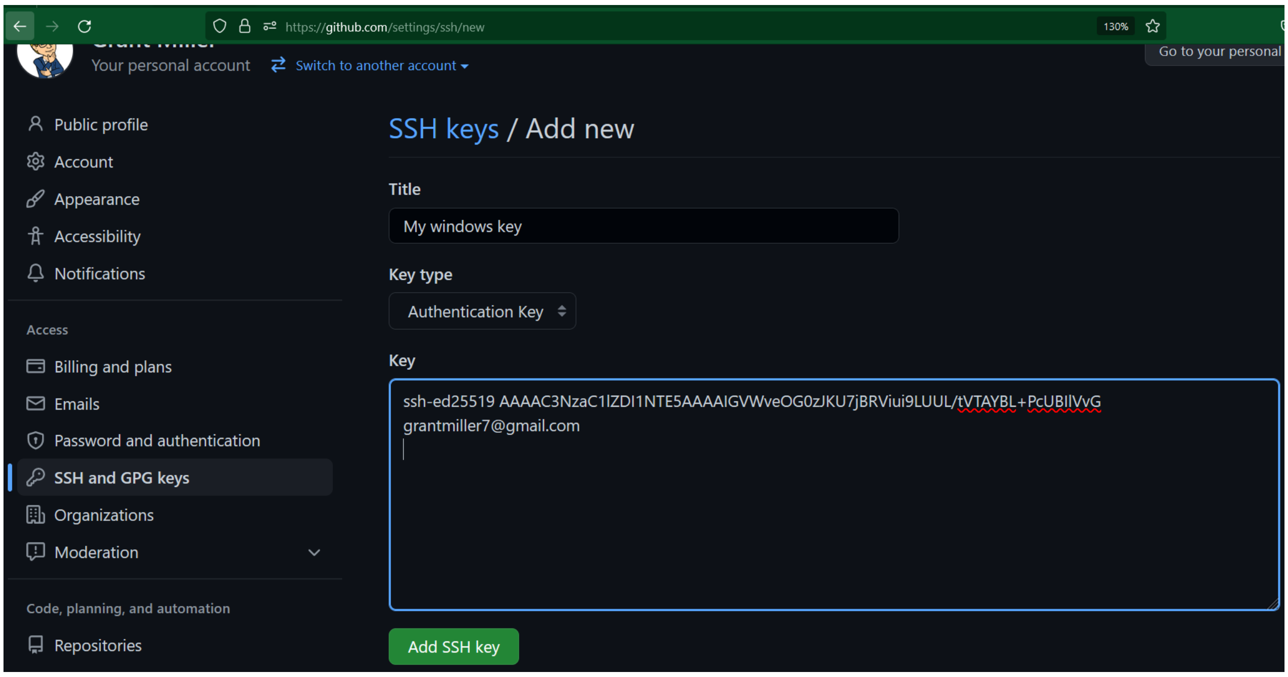
| Name | Date modified | Type | Size |
|---|---|---|---|
| id_ed25519 | 2022/10/31 08:20 | File | 1 KB |
| id_ed25519.pub | 2022/10/31 08:20 | Microsoft Publishe... | 1 KB |
| known_hosts | 2022/10/31 08:37 | File | 1 KB |
| known_hosts.old | 2022/10/31 08:37 | OLD File | 1 KB |

id_ed25519.pub - Notepad

```
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIGVWveOG0zJKU7jBRViui9LUUL/tVTAYBL+PcUBIlVvG grantmiller7@gmail.com
```

Ln 2, Col 1 100% Unix (LF) UTF-8

- Switch back to Github.com and paste the content in the `Key` Field



SSH keys / Add new

Title: My windows key

Key type: Authentication Key

Key:

```
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIGVWveOG0zJKU7jBRViui9LUUL/tVTAYBL+PcUBIlVvG
grantmiller7@gmail.com
```

- Click the green `Add SSH key` button in the lower left of your screen.

Useful reference

# Troubleshooting

## Test your connection to Github.com

Once you have followed the installation and configuation steps, the following should give a successful outcome:

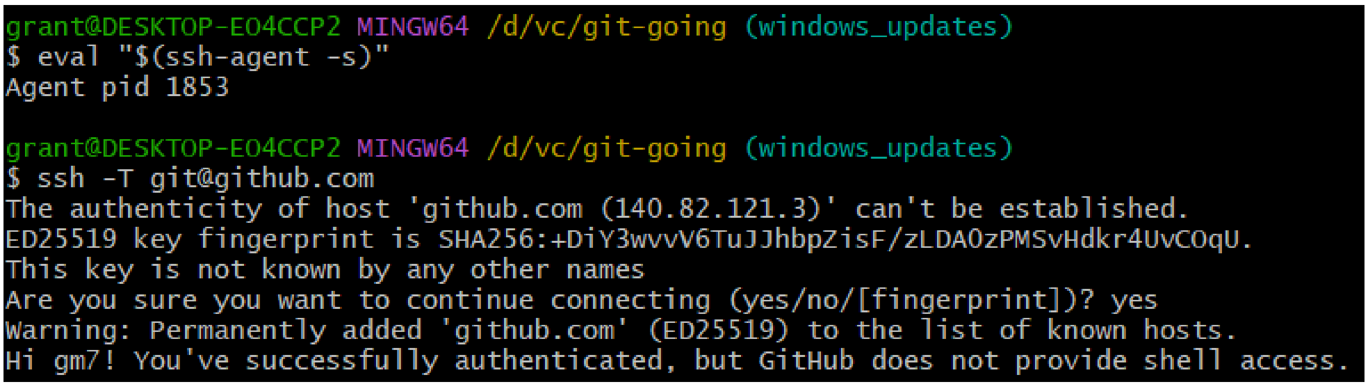
- Open `Git Bash` (Select `Git Bash` from the Windows Start Menu)
- Check that your open ssh client is running

```
eval "$(ssh-agent -s)"
```

- Test your connection to github.com

```
ssh -T git@github.com
```

Example:



```
grant@DESKTOP-EO4CCP2 MINGW64 /d/vc/git-going (windows_updates)
$ eval "$(ssh-agent -s)"
Agent pid 1853

grant@DESKTOP-EO4CCP2 MINGW64 /d/vc/git-going (windows_updates)
$ ssh -T git@github.com
The authenticity of host 'github.com (140.82.121.3)' can't be established.
ED25519 key fingerprint is SHA256:+DiY3wvvV6TuJJhbpZisF/zLDA0zPMSvHdkr4UvCOqU.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'github.com' (ED25519) to the list of known hosts.
Hi gm7! You've successfully authenticated, but GitHub does not provide shell access.
```

Note that the first time you connect to a server using SSH you will be asked if you want to continue connecting. You can safely type `yes` when connecting to github.com

### Unauthenticated error

If you receive an unauthenticated error, then try to re-add your key to the ssh-agent. The instructions are above, but here is a quick guide:

```
eval "$(ssh-agent -s)"
ssh-add ~/.ssh/id_ed25519
ssh-add -l
```

The last command will list the ssh keys that you have added to the agent. Try o test your connection again (see above).

If you are still not able to authenticate with github.com, then check that you added the correct key to github.com. Follow the steps above to add your public key to github.com.