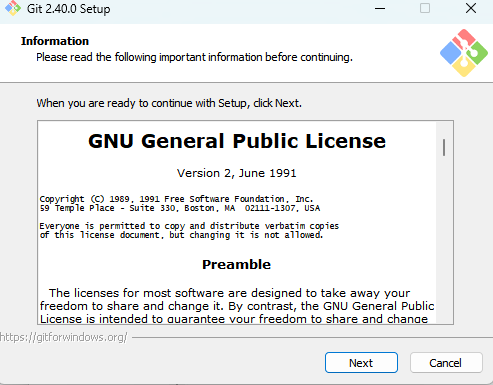
**第一、gib bash的安装**

**步骤1：首先进入官网找到下载链接https://gitforwindows.org**

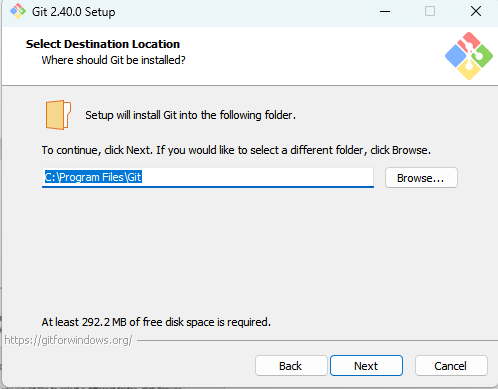


<https://github.com/git-for-windows/git/releases/download/v2.40.0.windows.1/Git-2.40.0-64-bit.exe>

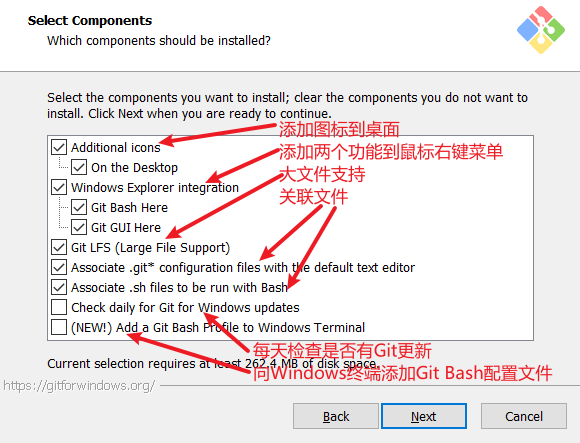
**步骤2：点击安装**



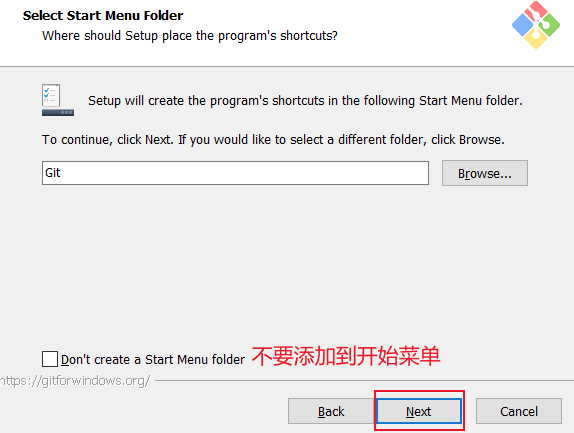
### ****步骤3：选择安装目录****



### ****步骤4：选择安装组件****

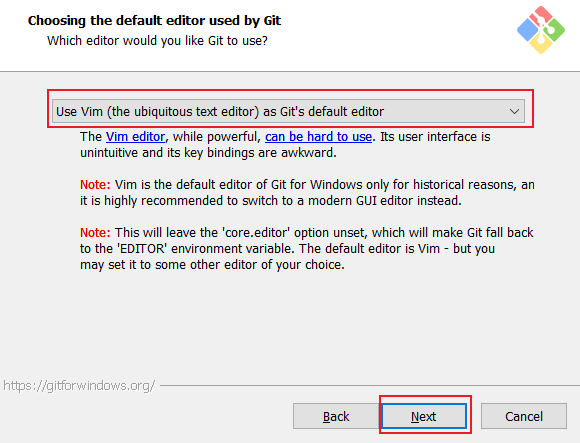


### ****步骤5：选择开始菜单文件夹****

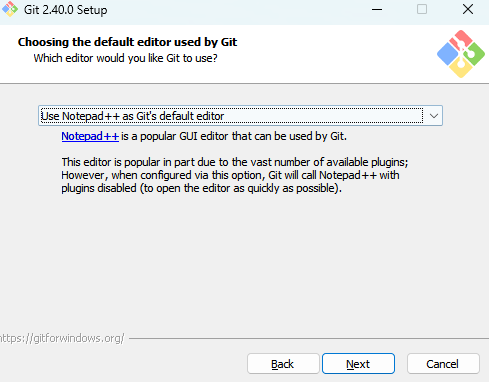


### ****步骤6：选择 Git 默认编辑器****

**Git 安装程序里面内置了 10 种编辑器，比如 Atom、Notepad、Notepad++、Sublime Text、Visual Studio Code、Vim 等等，默认的是 Vim ，选择 Vim 后可以直接进行到下一步，但是Vim是纯命令行，操作有点难度，需要学习。如果选其他编辑器，则还需要去其官网安装后才能进行下一步。下图为默认编辑器 Vim可直接点击Next到第七步。**



**如果不想用 Vim 当默认编辑器，换一个，比如 Notepad++ ，需要点击下面的蓝色字体"Notepad++"去其官网下载安装好才能进行下一步 next。**

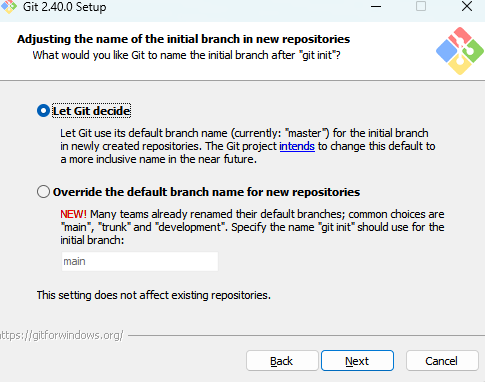


**安装后还要配置在我的电脑->属性->高级系统设置->高级->环境变量->系统变量->Path->编辑添加Notepad++的安装地址，如 C:\Program Files\notepad++，这样才能在 Git Bash 里面直接调用 Notepad++，点击Next到第七步。**

### ****步骤7. 决定初始化新项目(仓库)的主干名字(adjusting the name of the initial branch in new repositories)****

**第一种是让 Git 自己选择，名字是master，但是未来也有可能会改为其他名字；第二种是我们自行决定，默认是main，当然，你也可以改为其他的名字。一般默认第一种，点击Next到第八步。**

**注： 第二个选项下面有个NEW！，说很多团队已经重命名他们的默认主干名为main. 这是因为2020 年非裔男子乔治·弗洛伊德因白人警察暴力执法惨死而掀起的 Black Lives Matter(黑人的命也是命)运动，很多人认为master不尊重黑人，呼吁改为main.**

****

### ****步骤8. 调整path 环境变量****

**这步意思是指使用什么方式打开git？**

**第一种是仅从Git Bash使用Git。这个的意思就是你只能通过Git安装后的Git Bash来使用Git，其他的什么命令提示符等第三方软件都不行。**

**第二种是从命令行以及第三方软件进行 Git。这个就是在第一种基础上进行第三方支持，你将能够从Git Bash，命令提示符(cmd) 和 Windows PowerShell以及可以从 Windows 系统环境变量中寻找Git的任何第三方软件中使用Git，推荐使用这个。**

**第三种是从命令提示符使用Git和可选的 Unix 工具。选择这种将覆盖 Windows 工具，如 “ find 和 sort ”。只有在了解其含义后才使用此选项，适合比较懂的人折腾。**



**Adjusting your PATH environment(调整路径环境)**

**How would you like to use Git from the command line?(您希望如何使用命令行中的Git？)**

**use Git from Git Bash only（只使用Git Bash中的Git）**

**This is the most cautious choice as your PATH will not be modified at all.You wonly be able to use the Git command line tools from Git Bash.(这是最谨慎的选择，因为您的路径根本不会被修改，您将只能使用Git命令行工具Git Bash。)**

**git来自命令行，也来自第三方软件(Git from the command line and also from 3rd-party software）**

**(Recommended) This option adds only some minimal Git wrappers to yourPATH to avoid cluttering your environment with optional Unix tools.（(推荐)此选项只向您的PATH添加一些最小的Git包装器，以避免将您的环境与可选的Unix工具相混淆。）**

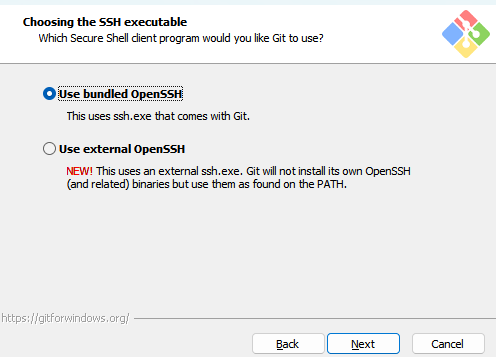
**You will be able to use Git from Git Bash, the Command Prompt and the Windows PowerShell as well as any third-party software looking for Git in PATH.（您将能够使用Git从Git Bash，命令提示符和WindovPowershell以及任何第三方软件寻找Git的路径。）**

**use Git and optional Unix tools from the Command Prompt(从命令提示符中使用Git和可选Unix工具)**

**Both Git and the optional Unix tools will be added to your PATH.(Git和可选的Unix工具都将添加到您的路径中。)**

**Warning: This will override Windows tools like “find” and “sort”. onlyuse this option if you understand the implications.(警告：这将覆盖Windows工具，如“查找”和“排序”。只有当你理解其含义时，才使用这个选项。)**

### ****步骤9. 选择SSH可执行文件****



**choosing the SSH executable(选择SSH可执行文件)**

**Which Secure Shell client program would you like Git to use?(您希望Git使用哪个安全的Shell客户端程序？)**

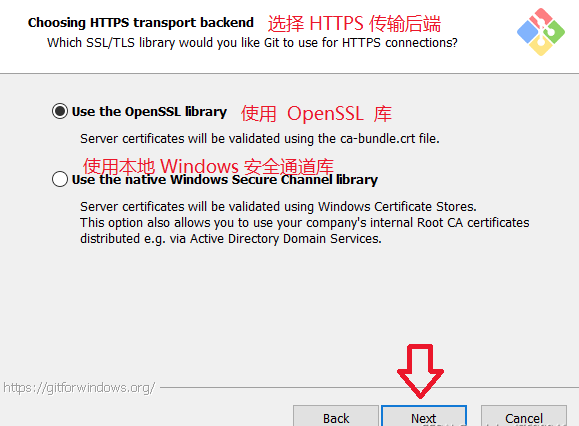
**use bundled opensSSH(使用捆绑开放SSH)**

**This uses ssh.exe that comes with Git.(它使用Git附带的ssh.exe。)**

**Use external opensSSH(使用外部操作系统SSH)**

**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.(新的！这将使用外部的ssh.exe.Git不会安装它自己的OpenSSH(和相关的)二进制文件，而是像在路径上找到的那样使用它们。)**

### ****步骤10. 选择HTTPS后端传输****



**Choosing HnIPS transport backend(选择HnIPS传输后端)**

**which SSL/TLS library would you like Git to use for HTTPS connections?(对于HTTPS连接，您希望Git使用哪个SSL/TLS库？)**

**use the openSSL library(使用OpenSSL库)**

**server certificates will be validated using the ca-bundle.crt file.(服务器证书将使用ca-bundle.crt文件进行验证。)**

**use the native Windows secure Channel library(使用本机Windows安全通道库)**

**Server certificates will be validated using Windows Certificate Stores.(服务器证书将使用Windows证书存储进行验证。)**

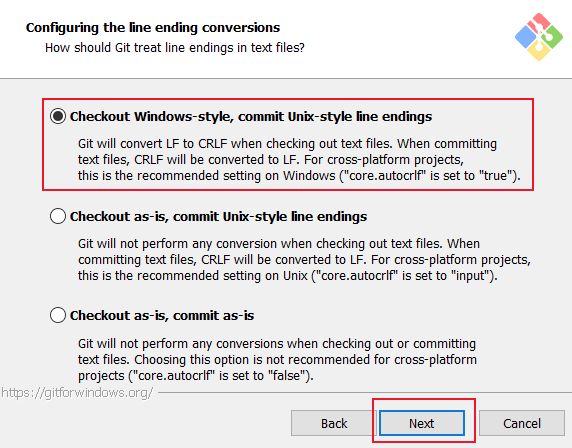
**This option also allows you to use your company’s internal Root CA certificatesdistributed e.g. via Active Directory Domain Services.(此选项还允许您使用公司的内部根CA证书，例如通过ActiveDirectory域服务分发的。)**

**这两种选项有什么区别呢？[git - What’s the difference between OpenSSL and the native windows Secure Channel library - Stack Overflow](https://stackoverflow.com/questions/62456484/whats-the-difference-between-openssl-and-the-native-windows-secure-channel-libr)**

**如果在具有企业管理证书的组织中使用 Git，则将需要使用安全通道。如果你仅使用 Git 来访问公共存储库（例如 GitHub ），或者你的组织不管理自己的证书，那么使用 SSL 后端（它们只是同一协议的不同实现）就可以了。**

**也就是说，作为普通用户，只是用 Git 来访问 Github、GitLab 等网站，选择前者就行了。点击next到第十一步。**

### ****步骤11. 配置行尾符号转换****



**Configuring the line ending conversions(配置行结束转换)**

**How should Git treat line endings in text files?(Git应该如何处理文本文件中的行尾?)**

**Checkout windows-style, commit Unix-style line endings(签出窗口样式，提交unix样式的行尾。)**

**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.autocrif" is set to “true”).(GIT将在签出文本文件时将LF转换为CRLF。提交文本文件时，CRLF将转换为LF。跨平台项目，这是在Windows上建议的设置(“core.autocrif”设置为“true”)。**

**checkout as-is, commit Unix-style line endings(以-is签出，提交unix样式的行尾。)**

**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”).(当签出文本文件时，git将不会执行任何转换。什么时候提交文本文件时，CRLF将转换为LF.or跨平台项目，这是Unix上的建议设置(“core.autocrlf”设置为“input”)。)**

**checkout as-is, commit as-is(按-原样结帐，按-原样提交)**

**Git will not perform any conversions when checking out or committingtext files.Choosing this option is not recommended for cross-platformprojects (“core.autocrlf” is set to “false”).(当签出或提交文本文件时，git将不会执行任何转换。对于跨平台项目，不建议选择此选项(“core.autocrlf”设置为“false”)。)**

**这三种选择分别是：**

**签出 Windows 样式,提交 Unix 样式的行结尾。**

**按原样签出，提交Unix样式的行结尾。**

**按原样签出，按原样提交。**

**那 Windows 样式和 Unix 样式到底有什么区别呢？**

**引用 《[GitHub 入门与实践](https://book.douban.com/subject/26462816/)》 [第 50 页内容](https://blog.csdn.net/mukes/article/details/115693833?ops_request_misc={"request_id":"164239270916780264034862","scm":"20140713.130102334.."}&request_id=164239270916780264034862&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~top_positive~default-1-115693833.pc_search_insert_es_download&utm_term=git%E5%AE%89%E8%A3%85&spm=1018.2226.3001.4187" \l "fn2)：GitHub 中公开的代码大部分都是以 Mac 或 Linux 中的 LF（Line Feed）换行。然而，由于 Windows 中是以 CRLF（Carriage Return+ Line Feed）换行的，所以在非对应的编辑器中将不能正常显示。**

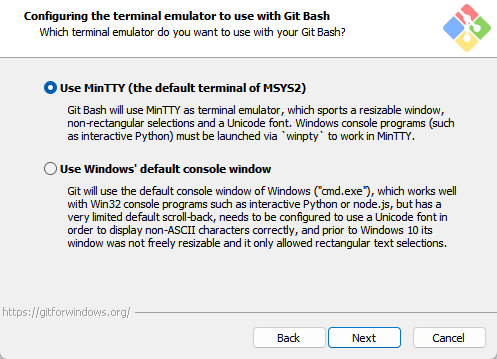
**Git 可以通过设置自动转换这些换行符。使用 Windows 环境的各位，请选择推荐的 “Checkout Windows-style，commit Unix-style line endings” 选项。换行符在签出时会自动转换为 CRLF，在提交时则会自动转换为 LF .**

**上面说 Mac 、Linux、Unix 的 Line Feed ，翻译过来就是换行符，用 “\n” 表示，换行符 “\n” 的 ASCII 值为10；**

**Windows 的是 Carriage Return+ Line Feed（回车+换行），用 “\r\n” 表示，回车符 “\r” 的 ASCII 值为13；**

**这上下两者是不一样的。  
我们现在的教程就是介绍怎么安装 Windows 版 Git，肯定选第一项啦。**

### ****步骤12. 配置终端模拟器以与 Git Bash 一起使用****



**Configuring the terminal emulator to use with Git Bash（配置用于Git Bash的终端模拟器）**

**which terminal emulator do you want to use with your Git Bash?（您想在Git Bash中使用哪个终端模拟器？）**

**Use MinITY (the default terminal of MsYs2)（使用MinITY(MsYs 2的默认终端)）**

**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. (Git Bash 将使用 MinTTY 作为终端仿真器，该仿真器具有可调整大小的窗口非矩形选择和 Unicode 字体。 Windows 控制台程序（例如交互式 Python）必须通过 “winpty” 启动才能在 MinTTY 中运行。)**

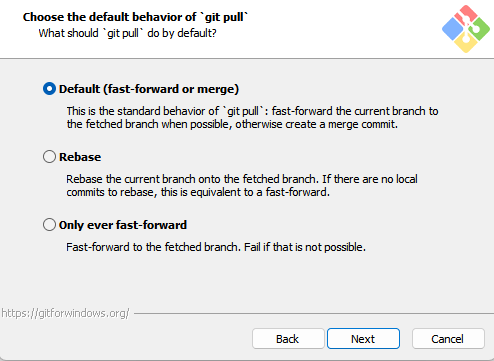
**Use windows’ default console window(使用windows的默认控制台窗口)**

**Git will use the default console window of Windows (“cmd.exe”), which works vwith Win32 console programs such as interactive Python or node.js, but has avery limited default scroll-back, needs to be configured to use a Unicode font inorder to display non-ASCII characters correctly, and prior to Windows 10 itswindow was not freely resizable and it only allowed rectangular text selections.(Git 将使用 Windows 的默认控制台窗口（“cmd.exe”），该窗口可与 Win32 控制台程序（例如交互式Python 或node.js）一起使用，但默认回滚非常有限，需要将其配置为使用 Unicode 字体才能正确显示非 ASCII 字符，并且在Windows 10 之前，其窗口不可随意调整大小，并且仅允许选择矩形文本。)**

**建议选择第一种，MinTTY 3功能比 cmd 多，cmd 只不过 比 MinTTY 更适合处理 Windows 的一些接口问题，这个对 Git 用处不大，除此之外 Windows 的默认控制台窗口（'cmd’）有很多劣势，比如 cmd 具有非常有限的默认历史记录回滚堆栈和糟糕的字体编码等等。**

**相比之下，MinTTY 具有可调整大小的窗口和其他有用的可配置选项，可以通过右键单击的工具栏来打开它们 git-bash 。点击next到第十三步。**

### ****步骤13.选择默认的 “git pull”行为****



**Choose the default behavior of git pull(选择git拉出的默认行为。)**

**what should git pull do by default?(默认情况下，“git pull”应该做什么？)**

**DefaulIt fasiforward .ormerge)(默认（快进或合并）)**

**This is the standard behavior of git pull : fast-forward the current branch tothe fetched branch when possible, otherwise create a merge commit.(这是 “git pull” 的标准行为：在可能的情况下将 当前分支 快进到 获取的分支，否则创建合并提交。)**

**Rebase(重基)**

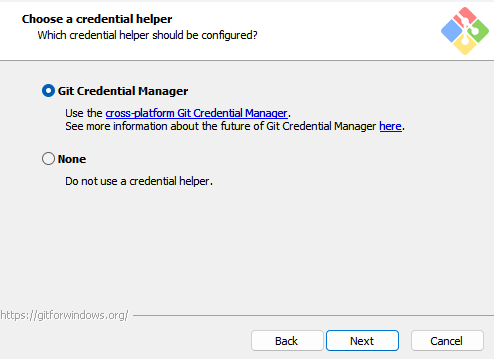
**Rebase the current branch onto the fetched branch. f there are no localcommits to rebase, this is equivalent to a fast-forward.(将当前分支重新定位到获取的分支上。不存在要重基的本地提交，这相当于快速转发.)**

**only ever fast-forward(只有永远快进)**

**Fast-forward to the fetched branch. Fail if that is not possible.(快进到抓取的树枝上。如果这是不可能的，那就失败。)**

**Q：“git pull”是什么意思呢？  
A：git pull就是获取最新的远程仓库分支到本地，并与本地分支合并**

### ****步骤14.选择一个凭证帮助程序****



**Choose a credential helper(选择凭据助手)**

**which credential helper should be configured?(应该配置哪个凭据助手？)**

**Git Credential Manager(Git凭证管理)**

**Use the cross-platform Git Credential Manager.(使用跨平台Git凭据管理器。)**

**See more information about the future of Git Credential Manager here.(请参阅有关Git凭证经理未来的更多信息。)**

**None(无**

**Do not use a credential helper.(不要使用凭据助手。)**

**一共两个选项：**

**Git 凭证管理**

**不使用凭证助手**

**第一个选项是提供登录凭证帮助的，Git 有时需要用户的凭据才能执行操作；例如，可能需要输入用户名和密码才能通过 HTTP 访问远程存储库（GitHub，GItLab 等等）。**

**Windows 的 Git 凭据管理器（Git Credential Manager）是在 Windows 上运行的基于 .NET Framework 的 Git 凭据帮助器。同样，适用于 Mac 和 Linux 的 Git 凭据管理器（Java GCM）是基于 Java 的 Git 凭据帮助器，仅可在 macOS 和 Linux 上运行。尽管这两个项目都旨在解决相同的问题（使用 Git 提供无缝的多因素 HTTPS 身份验证），但它们基于不同的代码库和语言，因此难以确保特性对等。**

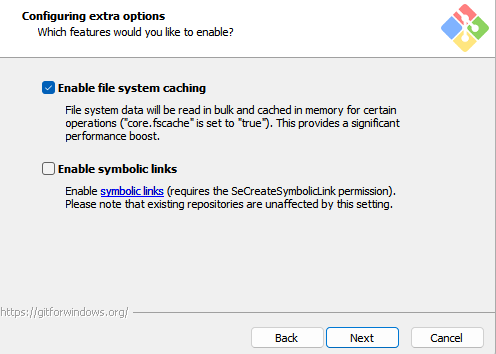
**Git Credential Manager Core（GCM Core）旨在用统一的代码库替换 GCM Windows 和 Java GCM，这在将来应该更容易维护和增强。**

**登录图如下(属于第一个选项的)**



**点击Next到十五步。**

### ****步骤15.配置额外的选项****



**Configuring extra options（配置额外选项）**

**which features would you like to enable?（您希望启用哪些功能？）**

**Enable file Svstem caching（启用文件缓存系统）**

**File system data will be read in bulk and cached in memory for certainoperations (“core.fscache” is set to “true”).This provides a significantperformance boost.（文件系统数据将大容量读取，并缓存在内存中以进行确定操作(“core.fscache”设置为“true”)，这将大大提高性能。）**

**Enable symbolic links(启用符号链接）**

**Enable symbolic links (requires the SeCreateSymbolicLink permission).（启用符号链接(需要SeCreateSymbolicLink权限)。）**

**Please note that existing repositories are unaffected by this setting.(请注意，现有的存储库不受此设置的影响。）**

**有两个选项：**

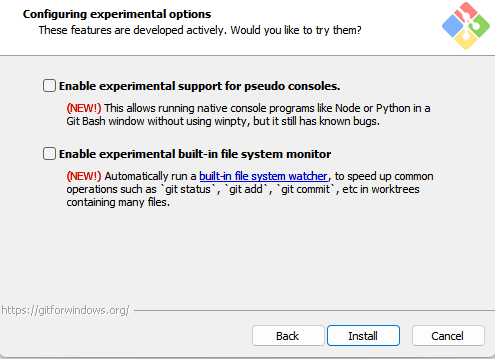
**启用文件系统缓存**

**启用符号链接**

**启用文件系统缓存就是将批量读取文件系统数据并将其缓存在内存中以进行某些操作，可以显著提升性能。这个选项默认开启。**

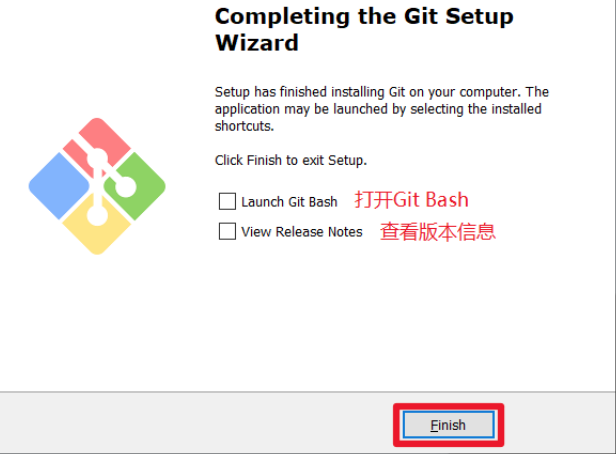
**启用符号链接，符号链接是一类特殊的文件，其包含有一条以绝对路径或者相对路径的形式指向其它文件或者目录的引用，类似于 Windows 的快捷方式，不完全等同类Unix（如 Linux）下的符号链接。因为该功能的支持需要一些条件，所以默认不开启。**

### ****步骤16.配置实验性选项****



**实验性功能，技术还不成熟，有已知的bug，不要勾选，点击右下角的install按钮，开始安装Git。**

### ****步骤17.点击 Finsh 按钮，Git 安装成功！****



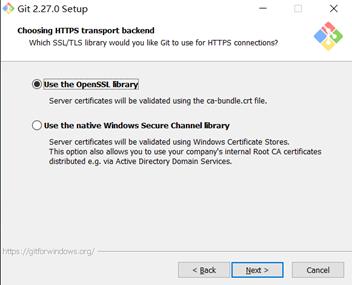
**这一步是提示我们git bash命令植入windows系统的深度，从上到下依次加深。**

**Use Git from Git Bash only表示我们只能在Git Bash中使用git命令；**

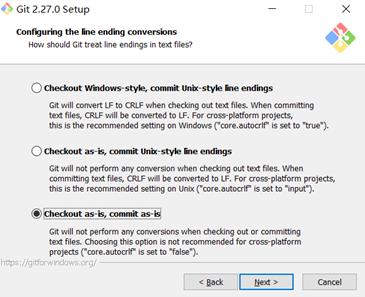
**Git from the command line and also from 3rd-party software表示我们除了可以在Git Bash中使用git命令外，还可以在像Cmd、PowerShell这样的第三方软件中使用git命令，也就是说会给windows系统配置git命令的环境变量，让你可以在cmd和powershell中使用git bash命令；**

**Use Git and optional Unix tools from the Command Prompt表示不仅会把git相关的命令加入windows的环境变量，同时也会把一些unix工具也加入系统变量中，例如会覆盖windows的find、sort命令行工具。除非是理解这个选项代表的含义，否则不建议选择该选项。**

**在这里，我们选择推荐选项，即“Git from the command line and also from 3rd-party software”。**



这里是让我们选择htts传输时使用的软件包，有SSL和TLS两种，我们这里使用默认的SSL。



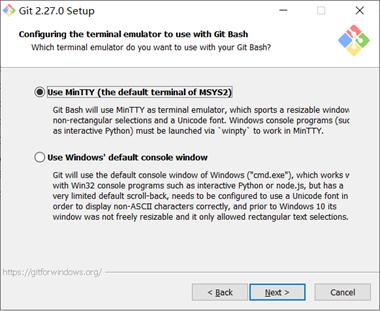
这里是让我们选择git检出文件时换行符的风格。

Checkout Windows-style,commit Unix-style line endings表示检出时转变为windows的换行符（即\r\n），提交时转变为unix的换行符（\n）；

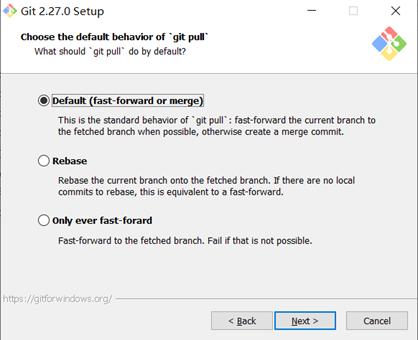
Checkout as-is,commit Unix-style line endings表示检出时不做任何的改变，原文件是什么就检出什么，但是提交时统一替换为unix的换行符；

Checkout as-is,commit as-is表示检出和提交时都不做什么的转变，原文件是什么就检出什么，原文件是什么就提交什么；

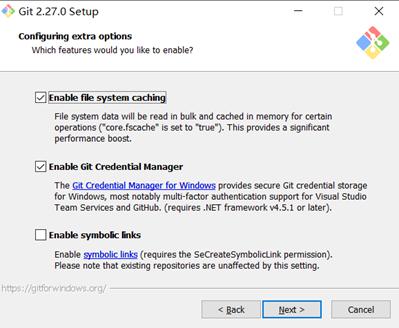
这里我们选择最后一种，检出了提交时不做什么的处理，直接提交，不过这种情况下，需要我们自己控制好文件的换行符，在编辑器中可以打开隐藏字符的展示，避免我们搞错了换行符。

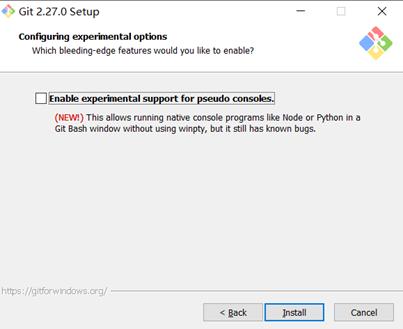


这里是让我们选择Git Bash命令行窗口的样式，默认是使用MinTTY，当然也可以选择使用windows的cmd窗口。



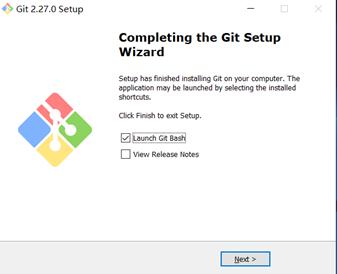
这里是让我们选择git pull命令的默认执行方式，默认是优先尝试fast-forward的方式合并分支，如果无法fast-forward进行merge操作。





这里是让我们选择添加一些附加的选项，包含一个实验性的新功能选项，感兴趣的话可以详细了解下，这里全部使用默认选项。

最后点击Install安装即可。





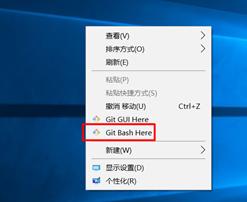
安装完成后，我们选择启动Git Bash，来看一下界面。

**第二、git bash的使用**

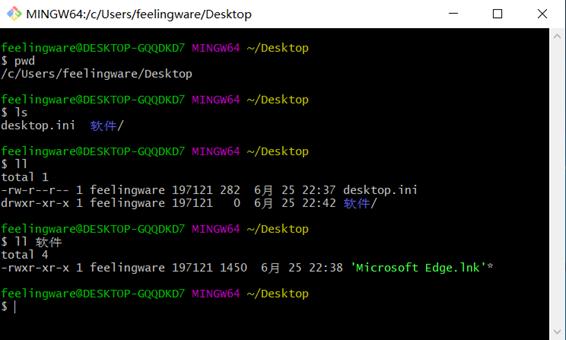
这里我们通过利用git bash在windows中生成ssh key来演示下gitbash的使用

参考：https://help.github.com/en/github/authenticating-to-github/generating-a-new-ssh-key-and-adding-it-to-the-ssh-agent

Git Bash安装完成后，我们可以在桌面或者任何目录下通过右键的方式进入git bash窗口



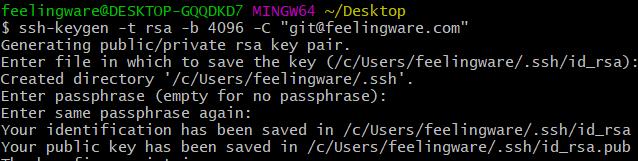
启动Git Bash窗口后我们简单的测试几个linux命令



然后我们来生成下ssh key

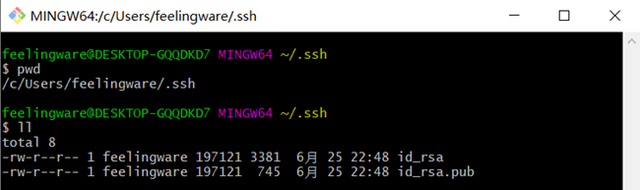
ssh-keygen -t rsa -b 4096 -C "**git@feelingware.com**"

保存位置默认，密码设置为空，直接敲3次回车就可以了



根据提示生成的密钥保存在用户目录下的.ssh文件夹中，我们去看一下





已经成功生成了密钥文件。