

# Git

## 11 总体介绍

### 学习目标

认识 git

### 概念

Git是目前世界上最先进的分布式版本控制系统（没有之一）

### 作用

用于团队开发的代码管理

### 诞生

Linus在1991年创建了开源的Linux，从此，Linux系统不断发展，已经成为最大的服务器系统软件了。

Linus虽然创建了Linux，但Linux的壮大是靠全世界热心的志愿者参与的，这么多人在世界各地为Linux编写代码，那Linux的代码是如何管理的呢？

事实是，在2002年以前，世界各地的志愿者把源代码文件发给Linus，然后由Linus本人通过手工方式合并代码！

你也许会想，为什么Linus不把Linux代码放到版本控制系统里呢？不是有CVS、SVN这些免费的版本控制系统吗？因为Linus坚定地反对CVS和SVN，这些集中式的版本控制系统不但速度慢，而且必须联网才能使用。有一些商用的版本控制系统，虽然比CVS、SVN好用，但那是付费的，和Linux的开源精神不符。

不过，到了2002年，Linux系统已经发展了十年了，代码库之大让Linus很难继续通过手工方式管理了，社区的弟兄们也对这种方式表达了强烈不满，于是Linus选择了一个商业的版本控制系统BitKeeper，BitKeeper的东家BitMover公司出于人道主义精神，授权Linux社区免费使用这个版本控制系统。

安定团结的大好局面在2005年就被打破了，原因是Linux社区牛人聚集，不免沾染了一些梁山好汉的江湖习气。开发Samba的Andrew试图破解BitKeeper的协议（这么干的其实也不只他一个），被BitMover公司发现了（监控工作做得不错！），于是BitMover公司怒了，要收回Linux社区的免费使用权。

Linus可以向BitMover公司道个歉，保证以后严格管教弟兄们，嗯，这是不可能的。实际情况是这样的：

Linus花了两周时间自己用C写了一个分布式版本控制系统，这就是Git！一个月之内，Linux系统的源码已经由Git管理了！牛是怎么定义的呢？大家可以体会一下。

Git迅速成为最流行的分布式版本控制系统，尤其是2008年，GitHub网站上线了，它为开源项目免费提供Git存储，无数开源项目开始迁移至GitHub，包括jQuery，PHP，Ruby等等。

历史就是这么偶然，如果不是当年BitMover公司威胁Linux社区，可能现在我们就没有免费而超级好用的Git了。

## 12 安装配置

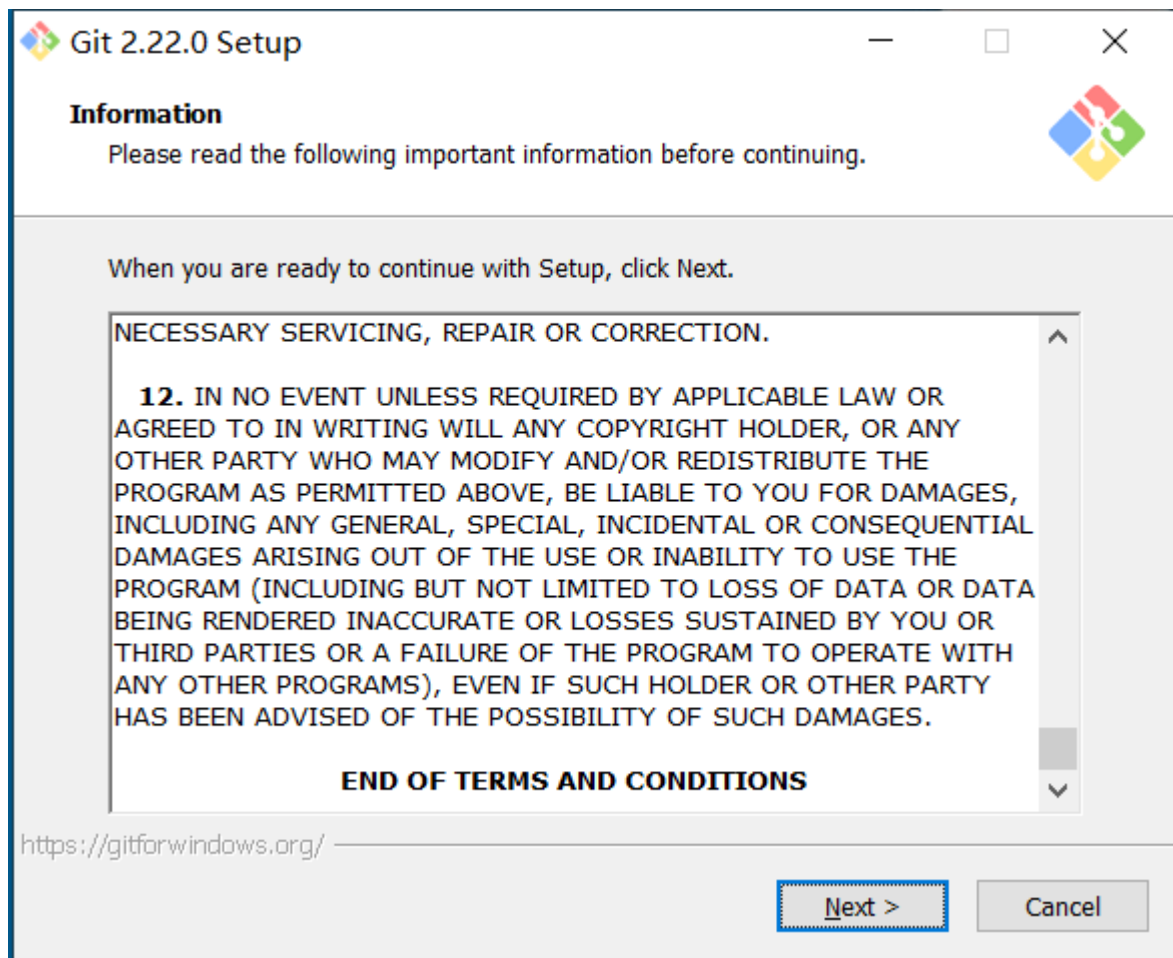
### 学习目标

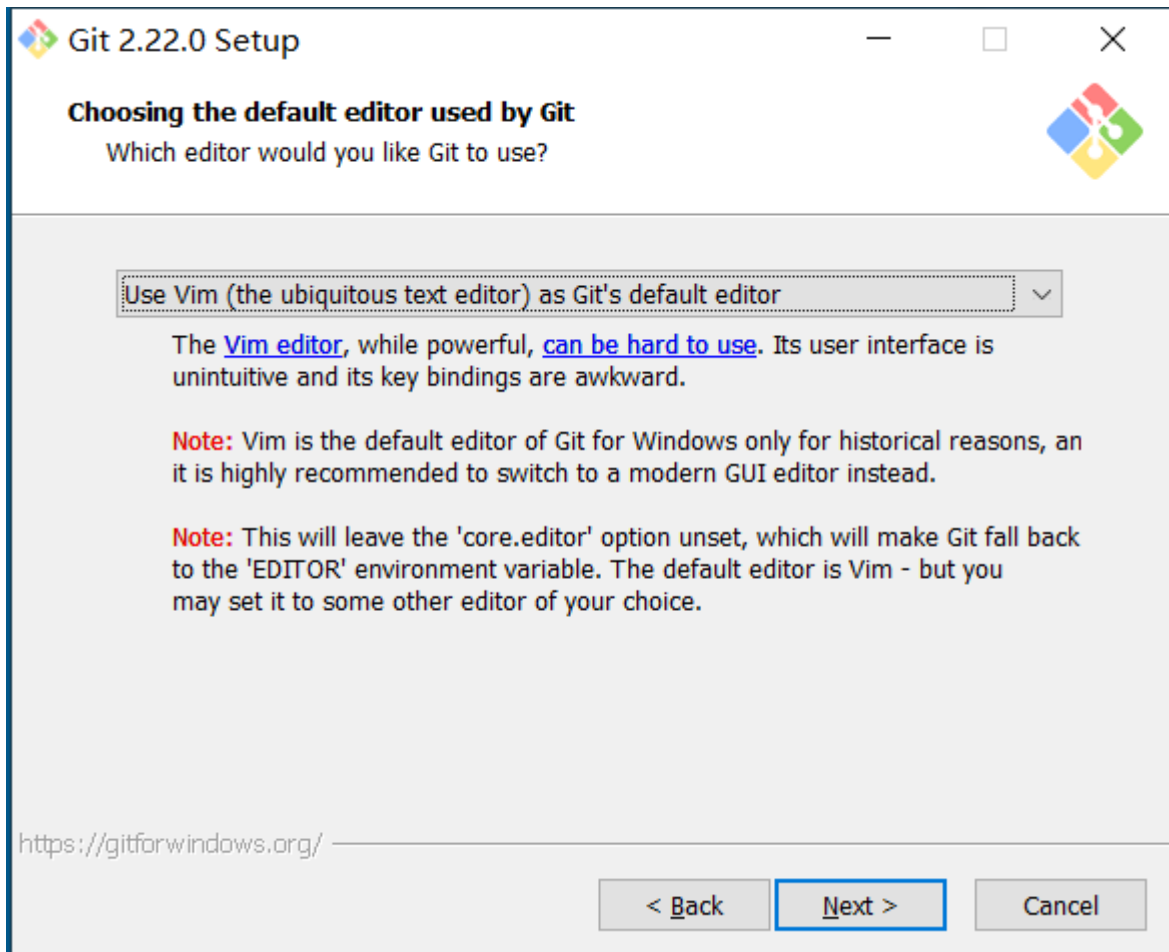
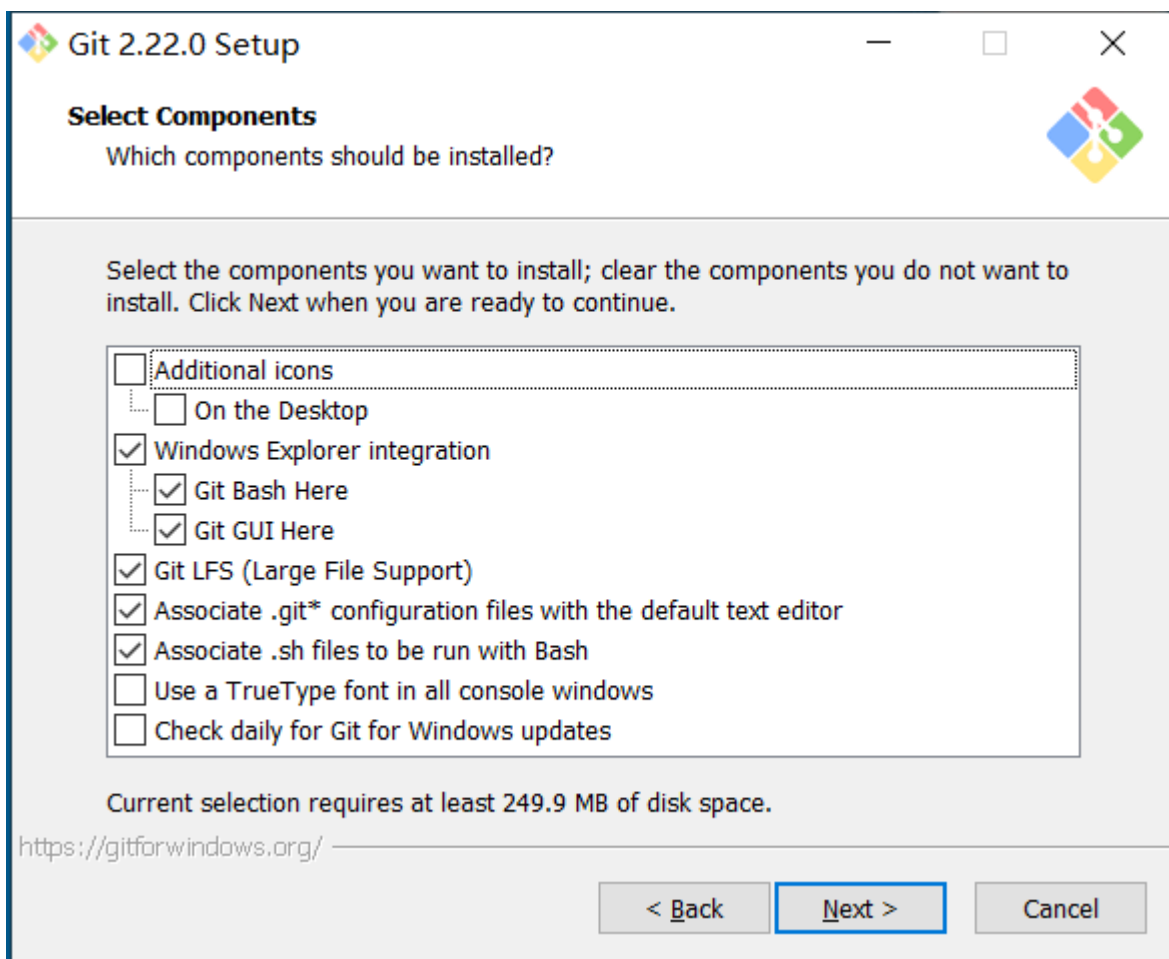
掌握git的安装配置

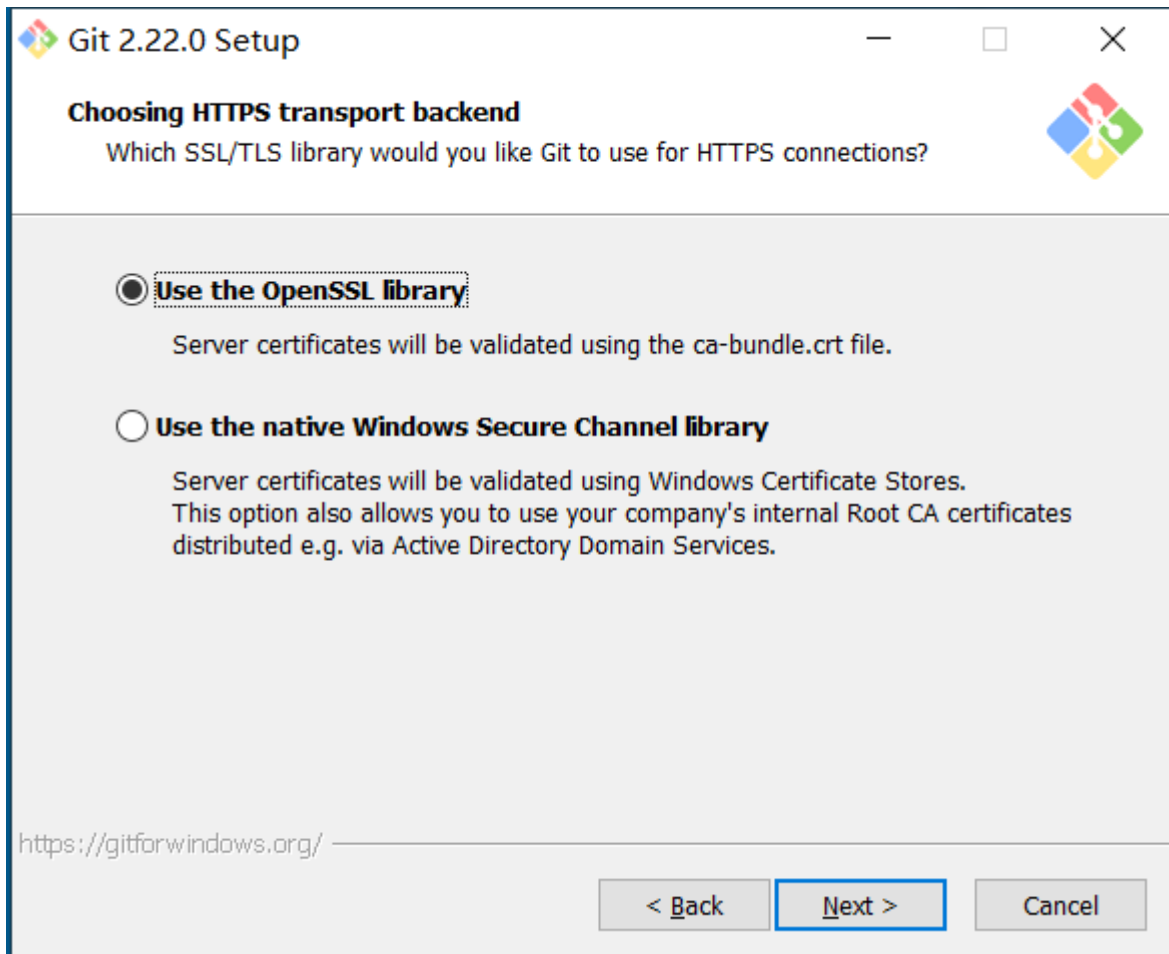
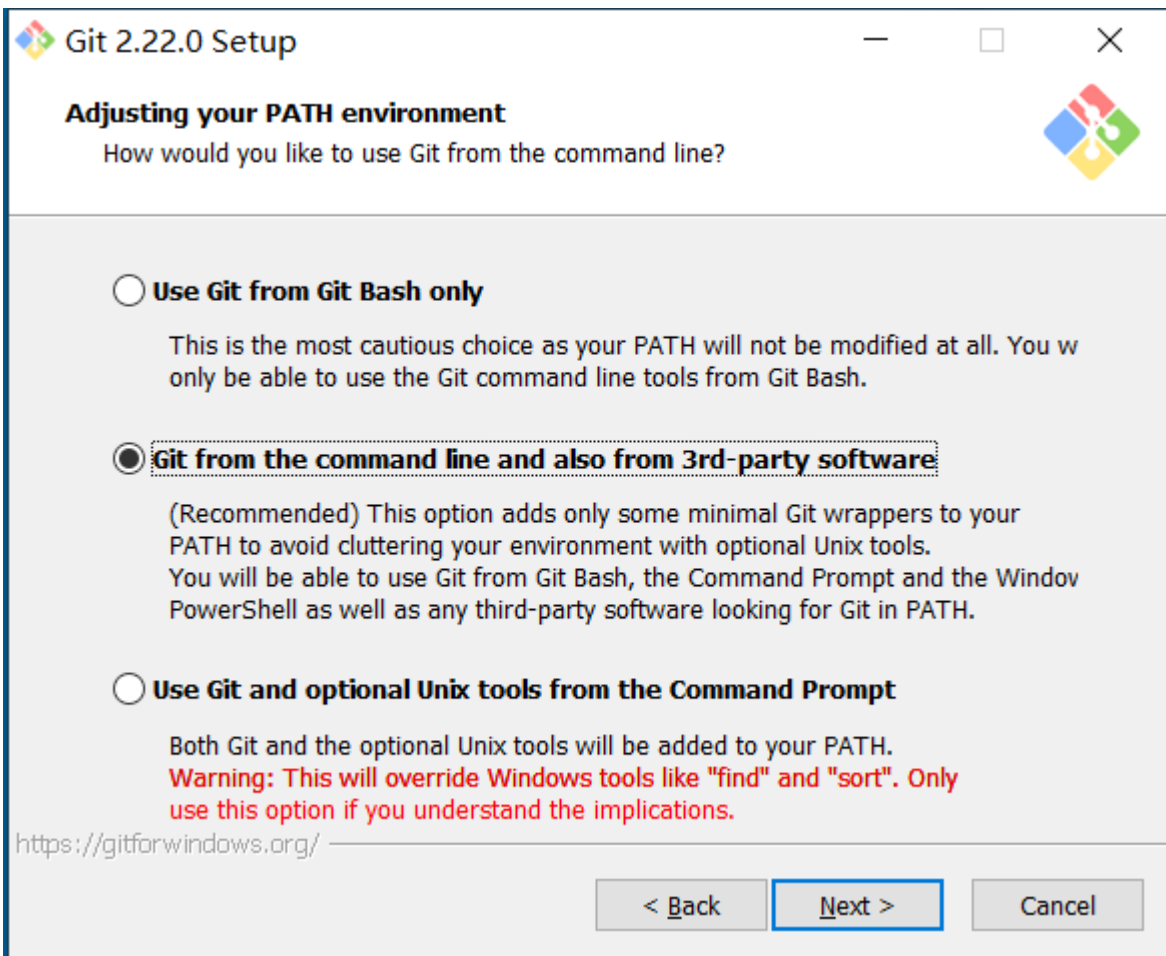
### 安装

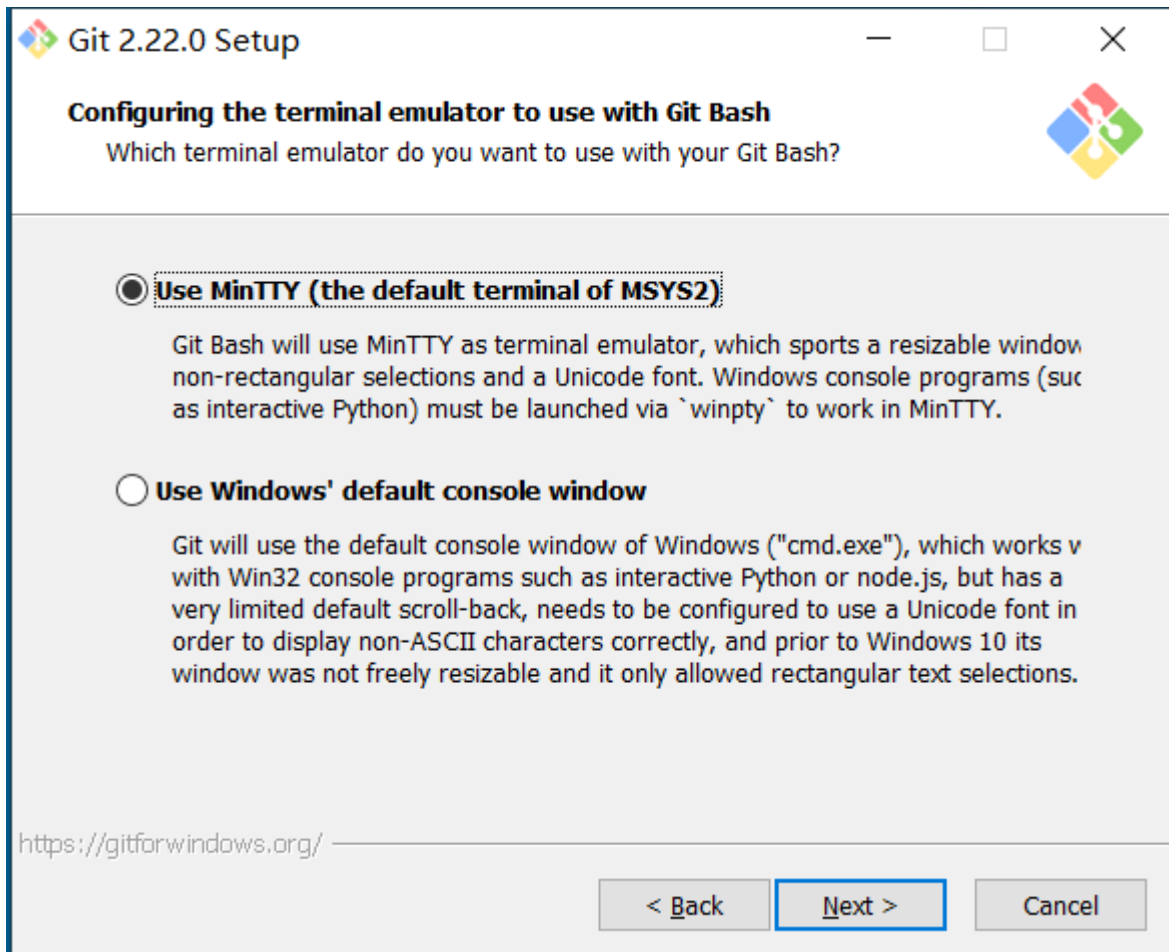
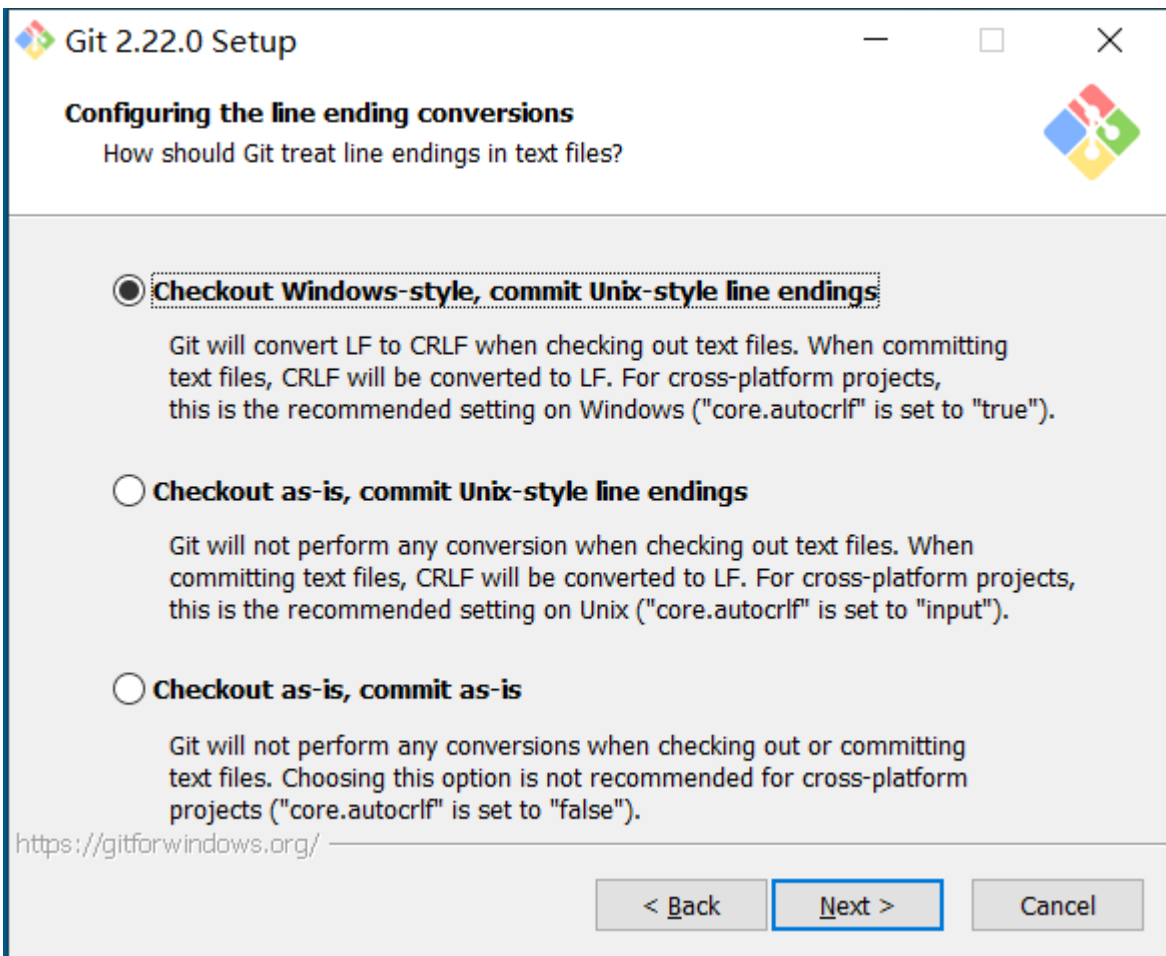
各系统版本下载地址: <https://git-scm.com/downloads>

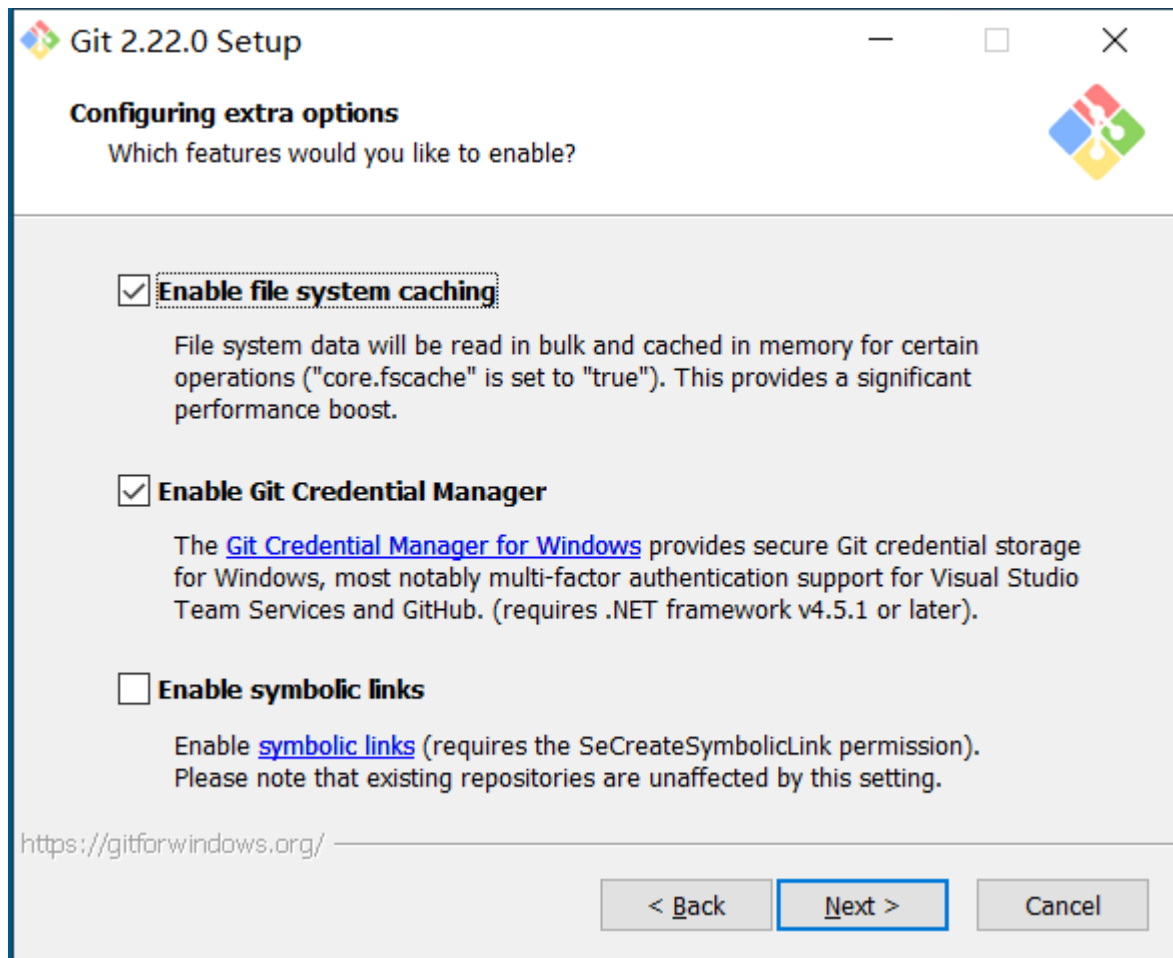
以 windows 为例: 双击 Git-2.22.0-64-bit.exe 进行安装











校验:

通过命令行输入 git, 没有提示不是内部或外部命令即可

## 配置

因为Git是分布式版本控制系统, 所以, 每个机器都必须自报家门: 需要配置你的名字和Email地址。

```
1 | git config --global user.name "Your Name"
2 | git config --global user.email "email@example.com"
```

校验:

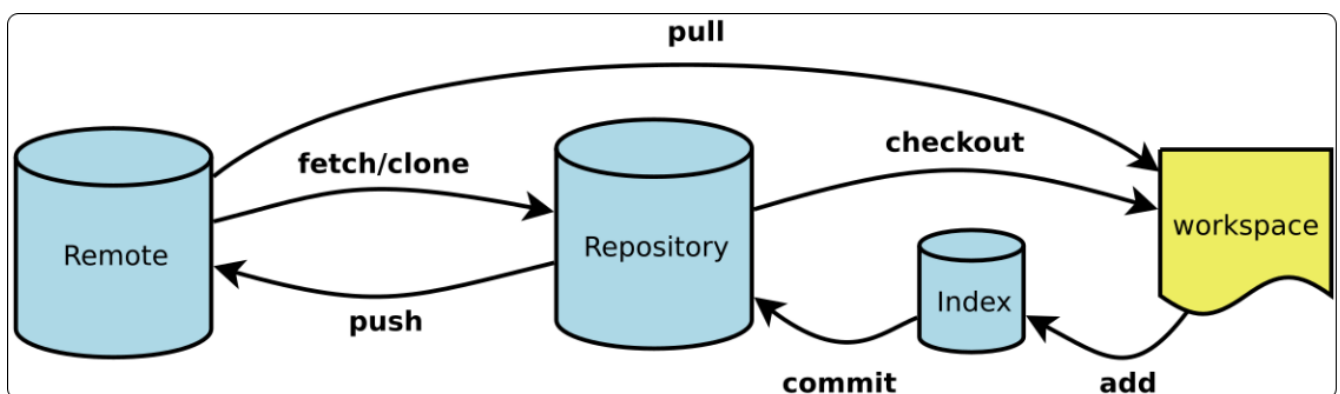
```
57769@JAY C:\Users\57769\Desktop
$ git config --list
core.symlinks=false
core.autocrlf=true
core.fscache=true
color.diff=auto
color.status=auto
color.branch=auto
color.interactive=true
help.format=html
rebase.autosquash=true
http.sslcainfo=d:/Program Files/Git/mingw64/ssl/certs/ca-bundle.crt
http.sslbackend=openssl
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
credential.helper=manager
user.name=liaoxingjie
user.email=57769717@qq.com
http.lowspeedlimit=0
http.lowspeedtime=999999
```

## 13 工作流

### 学习目标

了解git基本工作流

### 流程图解



workspace: 工作区, 就是你平时存放项目代码的地方

Index / Stage: 暂存区, 用于临时存放你的改动, 事实上它只是一个文件, 保存即将提交到文件列表信息

Repository: 本地仓库, 就是安全存放数据的位置, 这里面有你提交到所有版本的数据

Remote: 远程仓库, 托管代码的服务器

### 核心流程

## 1. workspace -> remote

在工作区中写代码, 提交到远程仓库

## 2. remote -> workspace

把远程仓库最新代码下载到本地, 开展工作

## 14 准备工作

### 学习目标

能够创建远端仓库

### 远程仓库有哪些

1. gitee
2. gitlab
3. github

### 创建github远程仓库

1. 进入首页: <https://github.com/>
2. 注册用户
3. 登录并创建远程仓库

## 15 workspace -> remote

### 学习目标

掌握提交代码到远程仓库的工作流

### 步骤

1. 创建工作区

```
1 | git init
```

2. 把工作区文件添加到暂存区

```
1 | git add .
```

只添加某一个文件: git add 文件名

3. 把暂存区文件提交到本地仓库

```
1 | git commit -m "xxx"
```

"xxx" 是关于本次提交的描述, 可以写具体修改的内容是什么


4. 把本地仓库的文件提交到远程仓库

1. 首次提交



```
1 | git push -u 远程仓库地址 master
```

-u 参数表示首次提交携带用户名和密码

 GitHub Login



# GitHub Login



Login



Cancel

Don't have an account? [Sign up](#)  
[Forgot your password?](#)

## 2. 非首次提交

```
1 | git push 远程仓库地址
```

### 解决远程仓库地址不便记的问题

添加远程仓库关联关系:

```
1 | git remote add origin 远程仓库地址
```

再次提交:

```
1 | git push origin
```

## 16 remote -> workspace

### 学习目标

掌握从远程仓库下载或更新代码的工作流

## 应用场景

1. 首次从远程仓库下载代码到本地仓库和工作区

```
1 | git clone 远程仓库地址
```

默认 clone master 分支的全部内容并在本地创建 master 分支

默认创建关联关系, 别名是 origin

2. 从远程仓库更新代码到本地仓库和工作区

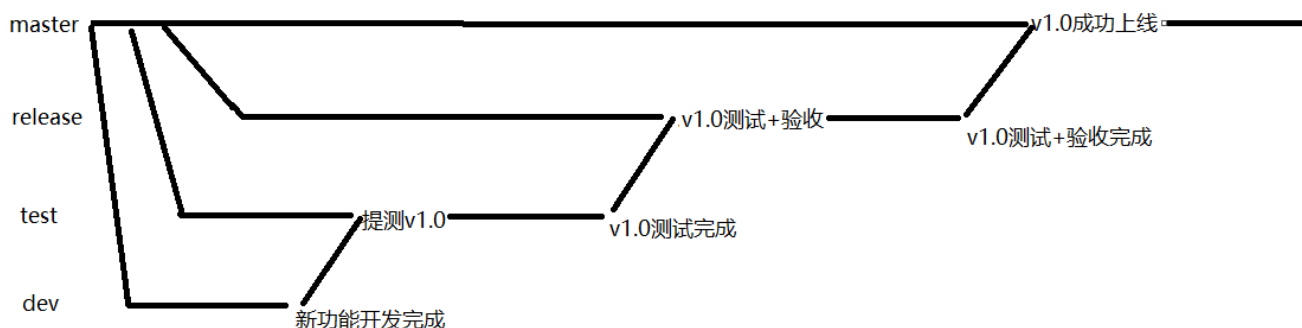
```
1 | git pull
```

## 17 分支管理

### 学习目标

了解实际工作中的分支管理流程及相关命令

### 流程示意图



### 相关命令

查看本地分支

```
1 | git branch
```

创建本地分支

```
1 | git branch 自定义分支名
```

切换本地分支

```
1 | git checkout 目标分支名
```

合并本地分支

```
1 | git merge 被合并到master的分支名
```

## 18 版本回退

### 学习目标

了解版本回退的用法

### 步骤

1. 列表展示当前git仓库的操作记录

```
1 | git reflog
```

2. 回退到某一版本

```
1 | git reset --hard 对应版本hash值
```

## 19 pycharm+git

### 学习目标

能够使用pycharm+git把代码推送到远程仓库

### GUI工具

概念:

GUI 的全称为 Graphical User Interface , 图形用户接口, 又称图形用户界面

列举:

1. 自带的
2. 小乌龟
3. SourceTree
4. pycharm

### pycharm+git的使用步骤

1. 配置 git, 让 pycharm 能使用 git

File -> Settings -> Version Control -> Git -> Path to Git executable

2. 初始化仓库, 相当于 git init

VCS -> Import into Version Control -> Create Git Repository

3. 配置忽略文件 .gitignore

1. 安装 .gitignore 插件

File -> Settings -> Plugins -> 搜索 .gitignore 并安装 -> 重启IDE

2. 新增或复制 .gitignore 文件

新增:

右键项目 -> New -> .ignore file -> .gitignore file -> Ignore File Generator -> 选 Example user template -> Generate

复制:

把 .gitignore 文件复制到项目根目录下

4. 添加文件到暂存区, 相当于 git add

右键文件 -> Git -> Add

5. 添加文件到本地仓库, 相当于 git commit -m "xxx"

右键文件 -> Git -> Commit File -> 填写 Commit Message -> Commit

6. 推送到远程仓库, 相当于 git push xxx

右键文件 -> Git -> Repository -> Push -> Define remote -> URL 填写远程仓库地址 -> Push