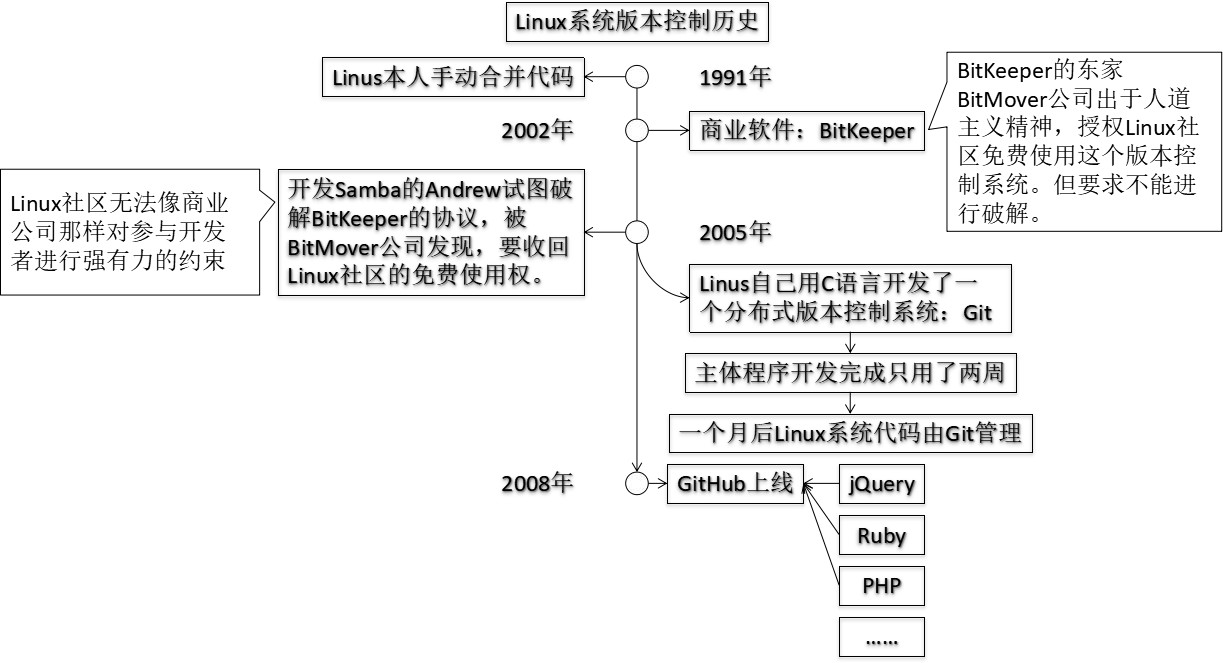
# 一.Git历史

同生活中的许多伟大事件一样，Git 诞生于一个极富纷争大举创新的年代。Linux 内核开源项目有着为数众广的参与者。绝大多数的 Linux 内核维护工作都花在了提交补丁和保存归档的繁琐事务上（1991

－2002年间）。到 2002 年，整个项目组开始启用分布式版本控制系统 BitKeeper 来管理和维护代码。

到 2005 年的时候，开发 BitKeeper 的商业公司同 Linux 内核开源社区的合作关系结束，他们收回了免费使用 BitKeeper 的权力。这就迫使 Linux 开源社区（特别是 Linux的缔造者 Linus Torvalds ）不得不吸取教训，只有开发一套属于自己的版本控制系统才不至于重蹈覆辙。他们对新的系统订了若干目标：



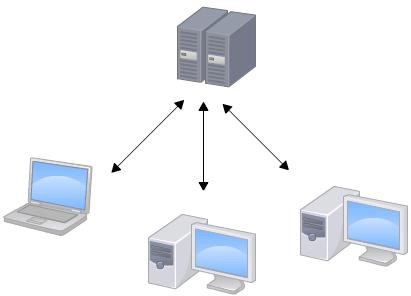
* 速度
* 简单的设计
* 对非线性开发模式的强力支持（允许上千个并行开发的分支）
* 完全分布式
* 有能力高效管理类似 Linux 内核一样的超大规模项目（速度和数据量）



# 二.Git与svn对比

## Svn

SVN是集中式版本控制系统，版本库是集中放在中央服务器的，而干活的时候，用的都是自己的电脑， 所以首先要从中央服务器哪里得到最新的版本，然后干活，干完后，需要把自己做完的活推送到中央服 务器。集中式版本控制系统是必须联网才能工作，如果在局域网还可以，带宽够大，速度够快，如果在 互联网下，如果网速慢的话，就郁闷了。

下图就是标准的集中式版本控制工具管理方式：

集中管理方式在一定程度上看到其他开发人员在干什么，而管理员也可以很轻松掌握每个人的开发权 限。

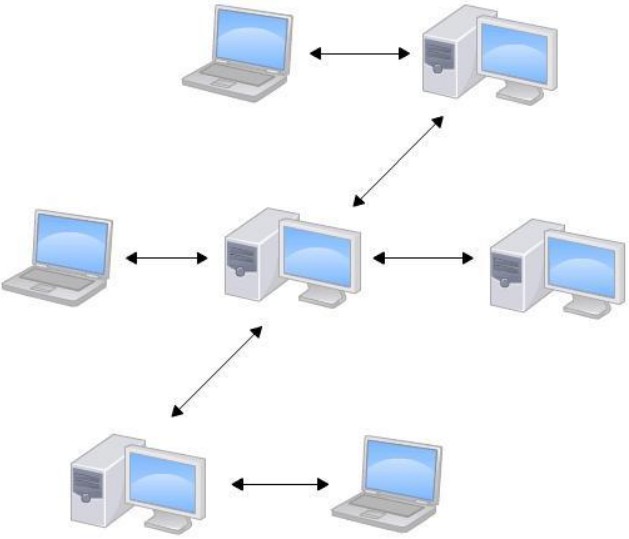
但是相较于其优点而言，集中式版本控制工具缺点很明显：

* + 1. 服务器单点故障
    2. 容错性差

## Git

Git(读音为/gɪt/。)是一个开源的**分布式版本控制系统**，可以有效、高速的处理从很小到非常大的项目版 本管理。 正因为是分布式版本控制系统，那么它就没有中央服务器的，每个人的电脑就是一个完整的版本库，这样，工作的时候就不需要联网了，因为版本都是在自己的电脑上。既然每个人的电脑都有一 个完整的版本库，那多个人如何协作呢？比如说自己在电脑上改了文件A，其他人也在电脑上改了文件A，这时，你们两之间只需把各自的修改推送给对方，就可以互相看到对方的修改了。

下图就是分布式版本控制工具管理方式：

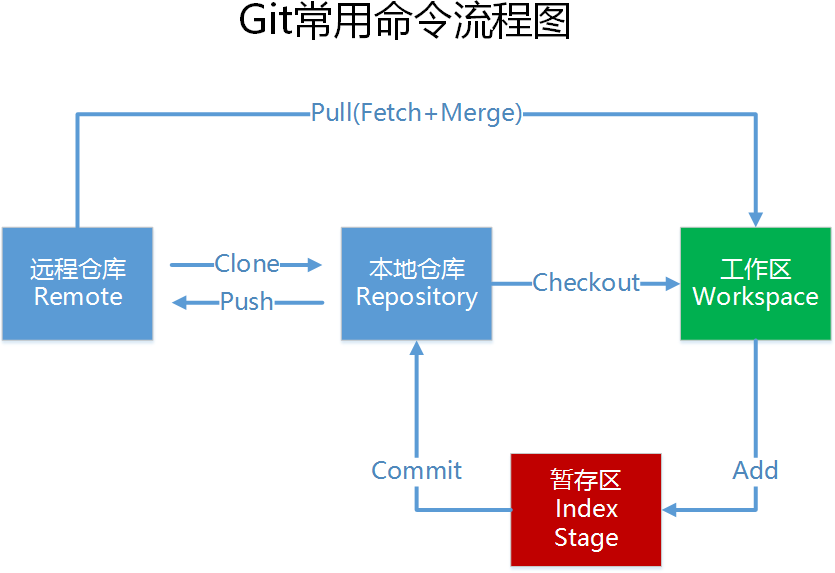


## Git工作流程

一般工作流程如下：

1．从远程仓库中克隆 Git 资源作为本地仓库。2．从本地仓库中checkout代码然后进行代码修改3．在提交前先将代码提交到暂存区。

1. 提交修改。提交到本地仓库。本地仓库中保存修改的各个历史版本。
2. 在修改完成后，需要和团队成员共享代码时，可以将代码push到远程仓库。 下图展示了 Git 的工作流程：



# 三.Git的安装

最早Git是在Linux上开发的，很长一段时间内，Git也只能在Linux和Unix系统上跑。不过，慢慢地有人 把它移植到了Windows上。现在，Git可以在Linux、Unix、Mac和Windows这几大平台上正常运行

了。由于开发机大多数情况都是windows，所以本教程只讲解windows下的git的安装及使用。

## 软件下载

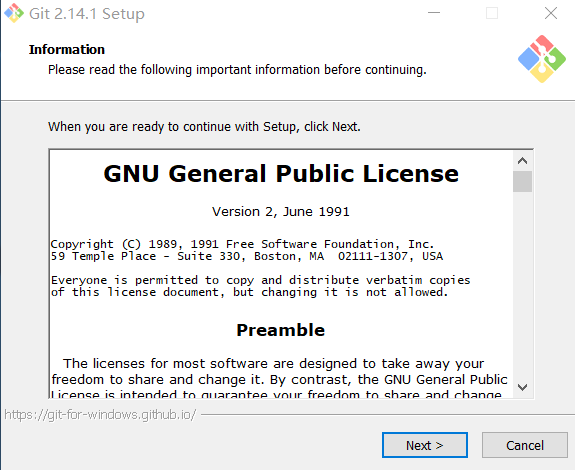
下载地址：<https://git-scm.com/download>

## 软件安装

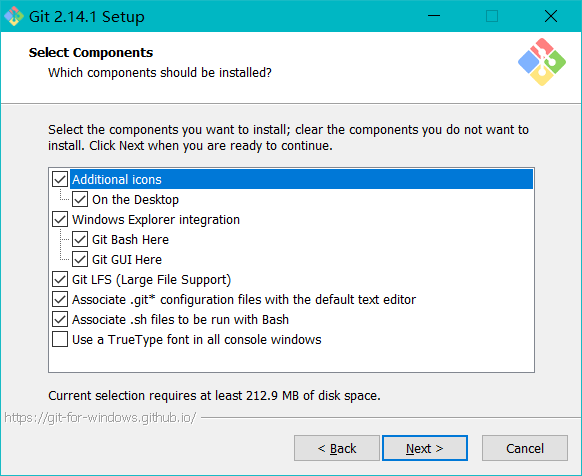
### 在安装包上面右键点击，选择以管理员身份运行



1. **安装包运行后，会出现Git的安装引导界面，第一个界面即是Git的授权信息介绍，阅读后可点击Next**

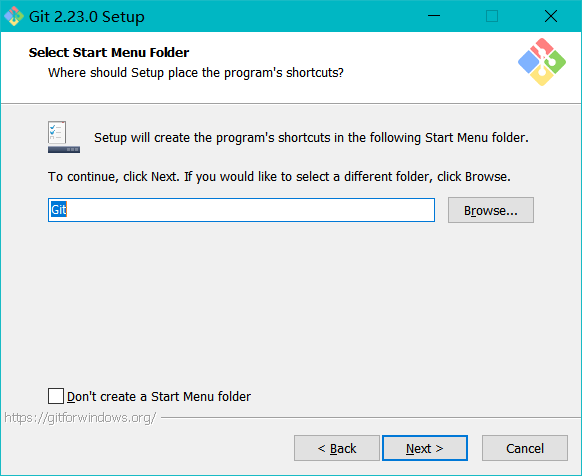
**按钮进行下一步安装**

1. **选择要安装的组件，建议仅选择默认勾选的组件，然后点击Next按钮**



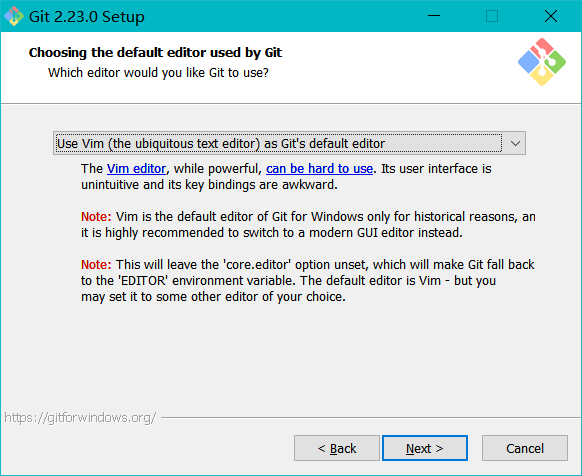
1. **设置开始菜单**

默认将创建一个名为Git的开始菜单文件夹，也可以点击Browse按钮选择创建新的开始菜单。建议默认 并点击Next按钮，如下图所示：

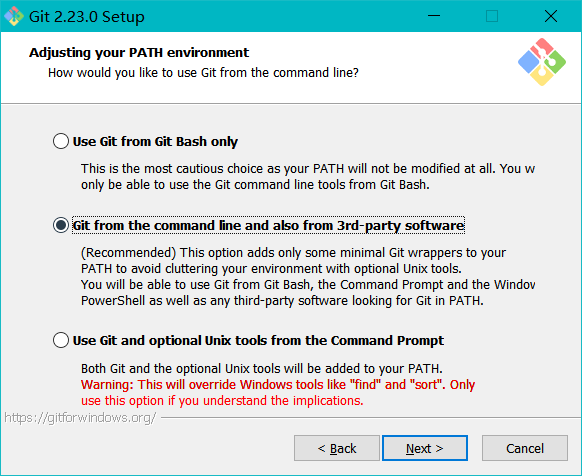


### 选择默认编辑器

选择Git默认使用的编辑器，一共有Nano、Vim、Notepad++、Visual Studio Code四种选择，默认已经选中Vim。建议选择默认选中的，然后点击Next按钮，如下图所示：

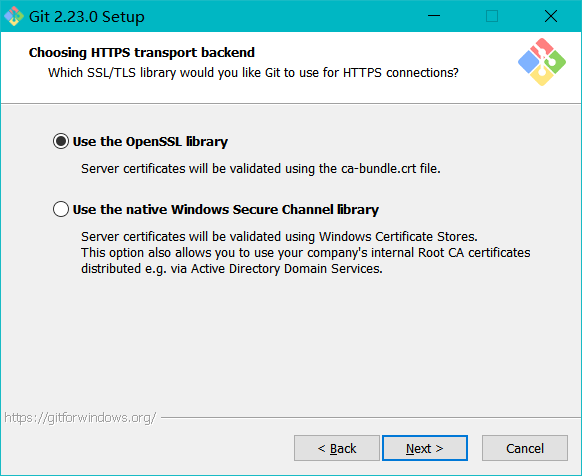


### 调整路径环境

路径环境设置与调整，默认选择第二种路径环境，即在Git Bash和Windows命令行窗口下使用Git。建议选择默认的并点击Next按钮，具体如下图所示：

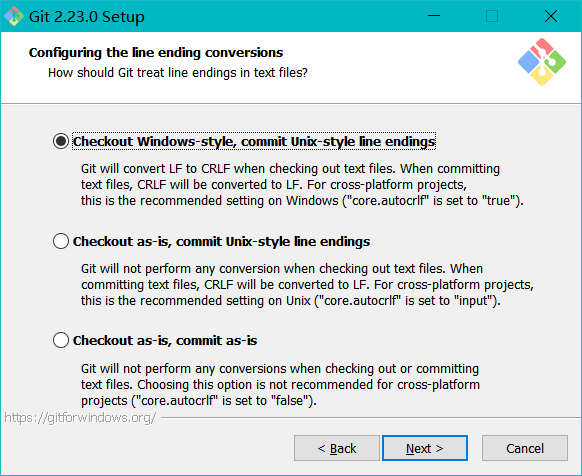
### 选择HTTPS传输端

选择HTTPS传输端，有OpenSSL library和Windows Secure Channel library两种选择。默认选择OpenSSL方式用于HTTPS连接，建议选择默认并点击Next按钮，如下图所示：

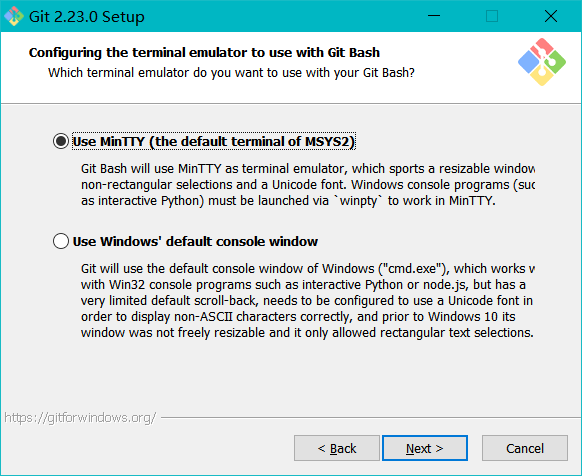


### 配置行尾转换

Git处理文本文件的行尾转换一共有三种方式可选，默认选择第一种，即检出时采用Windows风格，提 交时采用Unix风格。建议选择默认并点击Next按钮，如下图所示：



### 配置终端仿真程序

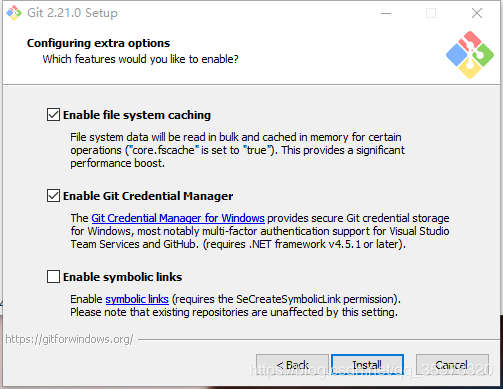
配置要与Git Bash一起使用的终端仿真程序，总共有MinTTY和Windows命令提示符窗口两种，默认选择MinTTY。建议选择默认并点击Next按钮，如下图所示

### 配置附加选项

配置其他的附加选项，默认启用文件系统缓存和Git凭据管理器。建议按照默认配置并点击Install按钮， 如下图所示

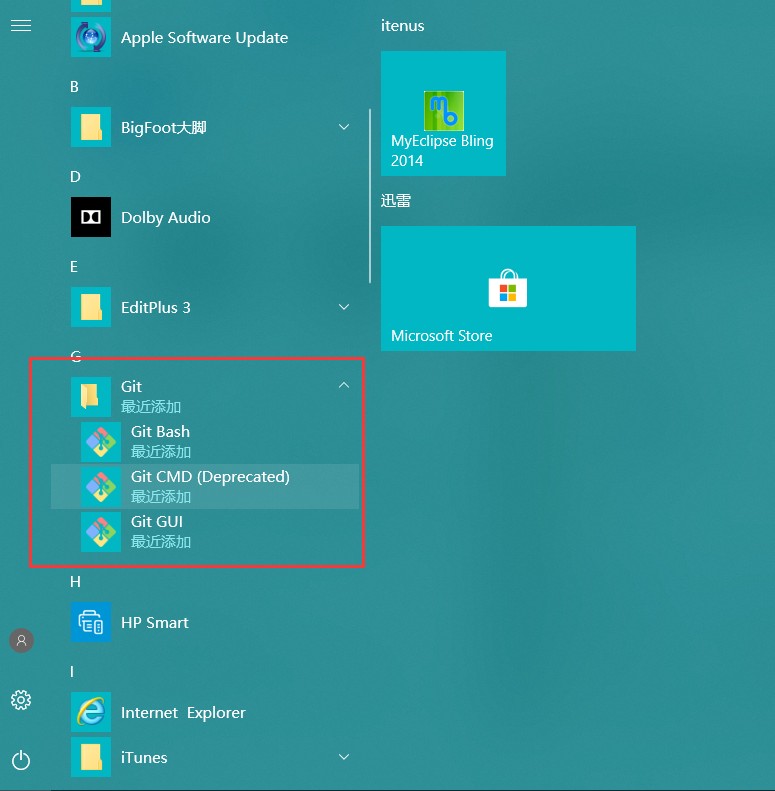
### 执行安装

在所有安装配置完成后点击Install按钮，Git将执行安装，如下图所示：

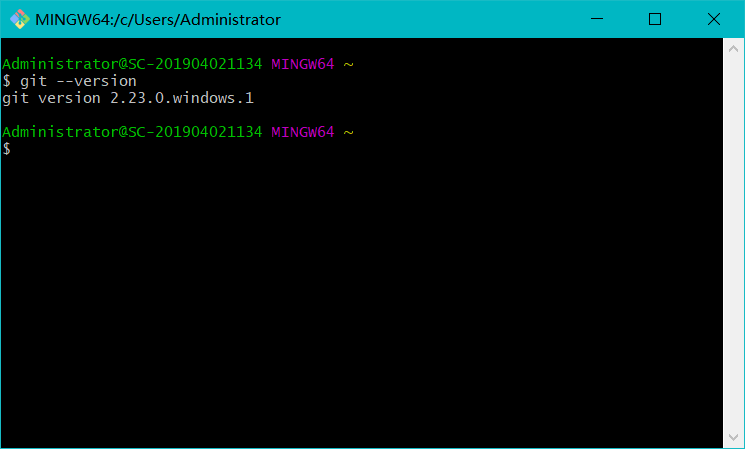


### 安装结果检查

在开始菜单能够看到Git及其下的Git Bash、Git CMD、Git GUI等菜单，则表示安装成功，如下图所示：



### 还可以打开Git Bash

并输入git --version命令查看已安装Git的版本号，如下图所示

# 四.Git概念梳理

## 代码托管中心：

任务：维护远程库局域网环境下

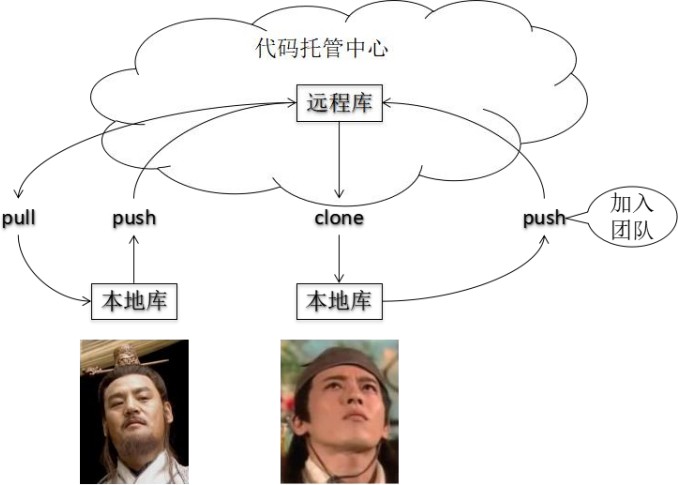
GitLab 服务器

外网环境下

GitHub

码云

## 本地库和远程库



* 1. **Git 命令行操作**

### 本地库初始化

命令： git init

注意：.git 目录中存放的是本地库相关的子目录和文件，不要删除，也不要胡乱修改。

### 设置签名

形式：

用户名：tom

Email 地址：[wangyanyi@apesource.com](mailto:goodMorning@atguigu.com) 作用：区分不同开发人员的身份

辨析：这里设置的签名和登录远程库(代码托管中心)的账号、密码没有任何关系

命令：

项目级别/仓库级别：仅在当前本地库范围内有效

git config user.name tom\_pro

git config user.email [wangyanyi@apesource.com](mailto:wangyanyi@apesource.com)

系统用户级别：登录当前操作系统的用户范围

git config--global user.name tom\_pro

[git config--global user.email wangyanyi@apesource.com](mailto:gitconfig--globaluser.emailwangyanyi@apesource.com)

级别优先级：

就近原则：项目级别优先于系统用户级别，二者都有时采用项目级别的签名如果只有系统用户级别的签名，就以系统用户级别的签名为准

二者都没有不允许

### 状态查看

查看工作区、暂存区状态命令： git status

### 添加

将工作区的“新建/修改”添加到暂存区命令： git add [file name]

### 提交

将暂存区的内容提交到本地库

命令： git commit -m "commit message" [file name]

### 查看历史记录

命令： git log

**创建远程库地址别名**

git remote -v 查看当前所有远程地址别名

git remote add [别名] [远程地址]

### 推送

git push [别名] [分支名master]

### 克隆

git clone [远程地址]

### 再一次使用新用户push自己新编辑的代码

会发现没有权限，在码云邀请同学

**仓库地址**：https://gitee.com/userwangtwo/java-ee-2109db.git

**申请成员：<https://gitee.com/userwangtwo/java-ee-2109db/invite_link?invite=a93e048ffbea98ed6abdff2aeb02e44f0553e3f71279db24b7993663152babc5f47e3f012b915e965f318cd36bbddc3a>**

