

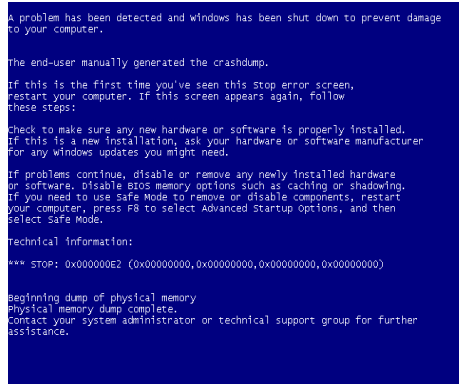
Git与代码版本控制

1. 版本控制系统简介
2. 事前准备
3. Git常用指令以及工作原理
4. 学习参考
5. 作业

版本控制系统简介

版本控制系统 (Version Control System, 简称 VCS)

- 让项目开发者在分工合作时避免了不必要的重复与冲突
- 如果遇到问题, 也可以及时回退到之前的版本。



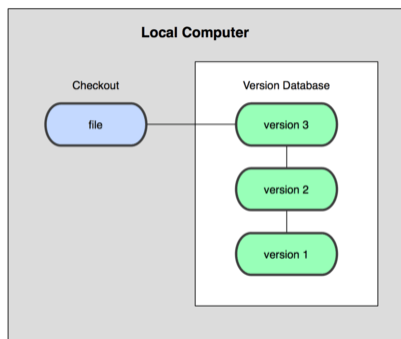
Blue screen of death

版本管理的演变

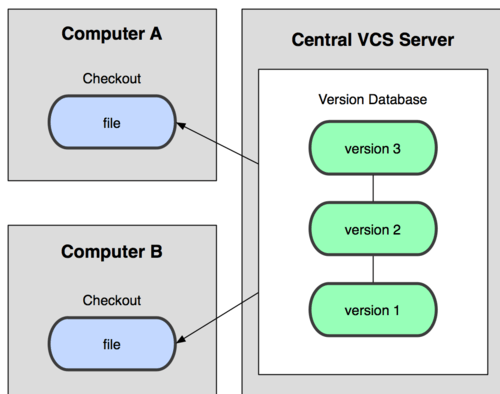


VCS出现前

- 用目录拷贝区别不同的版本
- 公共的文件很容易被覆盖
- 成员沟通成本高，代码集成效率低



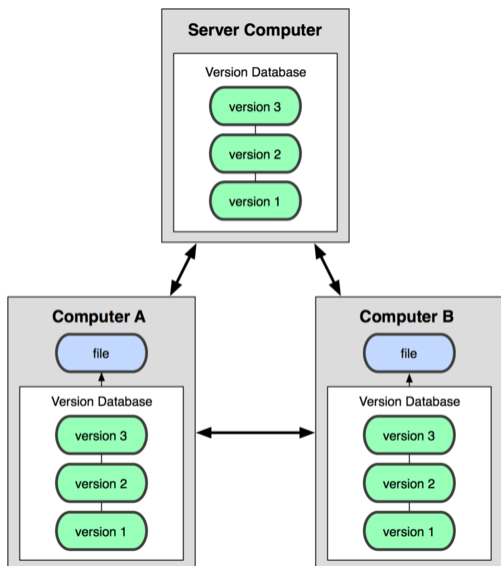
版本管理的演变



集中式VCS

- 集中的版本管理服务器
- 文件版本管理和分支管理
- 集成效率有明显的提高
- 客户端必须时刻和服务器相连

版本管理的演变



分布式VCS

- 服务端和客户端都有完整的版本库
- 脱离服务端，客户端照样可以管理版本
- 查看版本历史、版本比较等多数操作都不需要访问服务器，比集中式VCS更能提高版本管理效率

版本管理的演变



Git Creator Linus Torvalds



- ◆ 最优的存储能力
- ◆ 非凡的性能
- ◆ 开源
- ◆ 很容易做备份
- ◆ 支持离线操作
- ◆ 容易定制工作流程





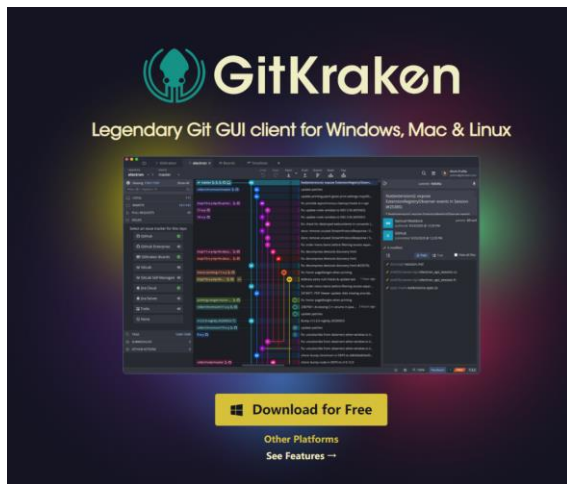
事先准备

为了使用Git和Github,你首先需要:

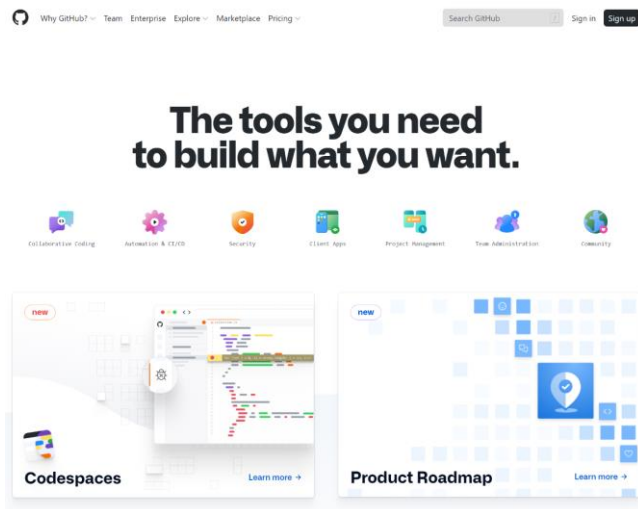
- 安装了Git的电脑 (详见 [Git 下载网址](#)) 。
- 用于使用 Git 的工具。你可以使用一个 [Git 图形操作界面客户端](#) (我们推荐 GitHub Desktop、Source Tree 或者 Git Kraken) 或者也可以直接使用命令行, 这取决于你的喜好。
- [GitHub 账户](#)。



The image shows the official Git website. At the top, it says "git --distributed-even-if-your-workflow-isnt". Below this, there's a search bar and a paragraph describing Git as a free and open source distributed version control system. To the right, there's a diagram showing a branching model with boxes representing commits and arrows representing branches. Below the text, there are sections for "About", "Documentation", "Downloads", and "Community". The "Downloads" section highlights the latest source release as 2.29.0. At the bottom, there's a section titled "Companies & Projects Using Git" with logos for Google, Facebook, Microsoft, LinkedIn, Netflix, and others.



The image shows the GitKraken website. It features the GitKraken logo at the top, followed by the text "Legendary Git GUI client for Windows, Mac & Linux". Below this, there's a screenshot of the GitKraken application interface, which displays a complex graph of commit history and branches. At the bottom, there's a yellow button that says "Download for Free" and a link to "Other Platforms See Features --".



The image shows the GitHub website. At the top, there's a navigation bar with links for "Why GitHub?", "Team", "Enterprise", "Explore", "Marketplace", and "Pricing". Below this, there's a search bar and a "Sign in" button. The main heading reads "The tools you need to build what you want." Below this, there are icons for various GitHub features: Collaborative Coding, Automation & CI/CD, Security, Cloud Apps, Project Management, Team Administration, and Community. At the bottom, there are two sections: "Codespaces" and "Product Roadmap", each with a "Learn more --" link.

事先准备

不同的操作系统安装Git:

Linux(Debian)

```
$ sudo apt-get install git
```

Linux(Fedora)

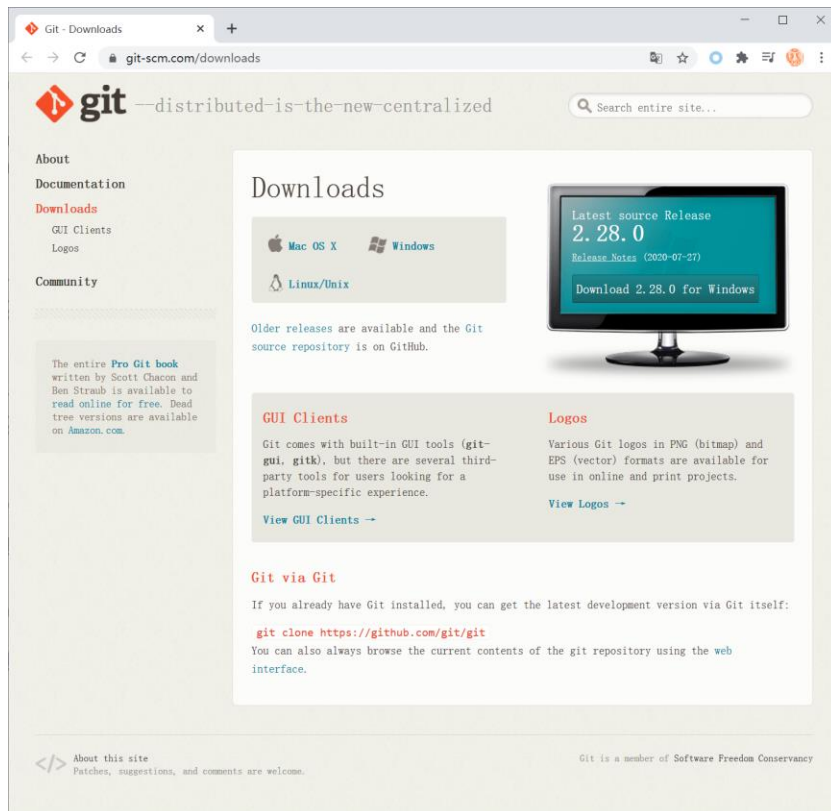
```
$ sudo yum install git
```

Mac

<http://git-scm.com/download/mac>

Windows

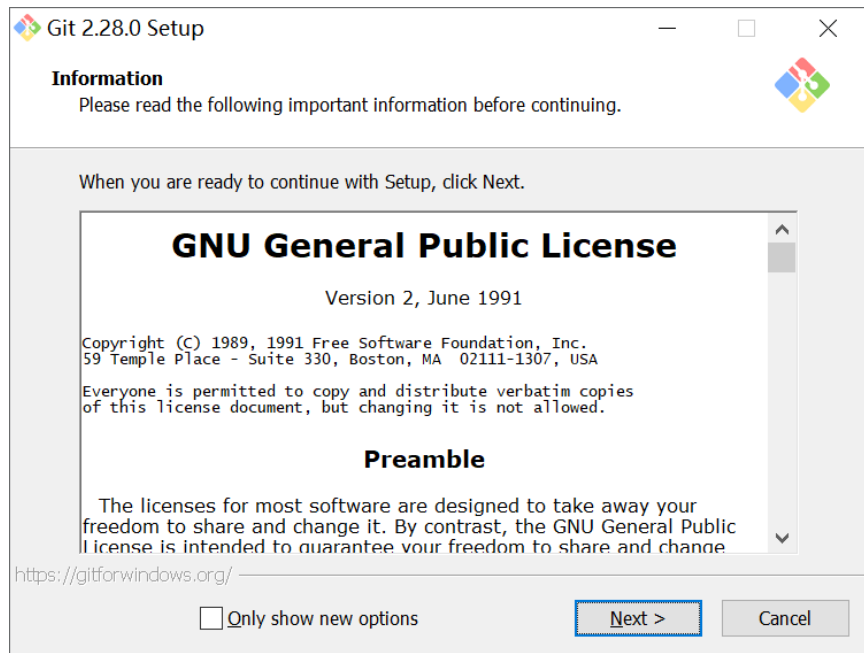
<http://git-scm.com/download/win>



windows下安装:

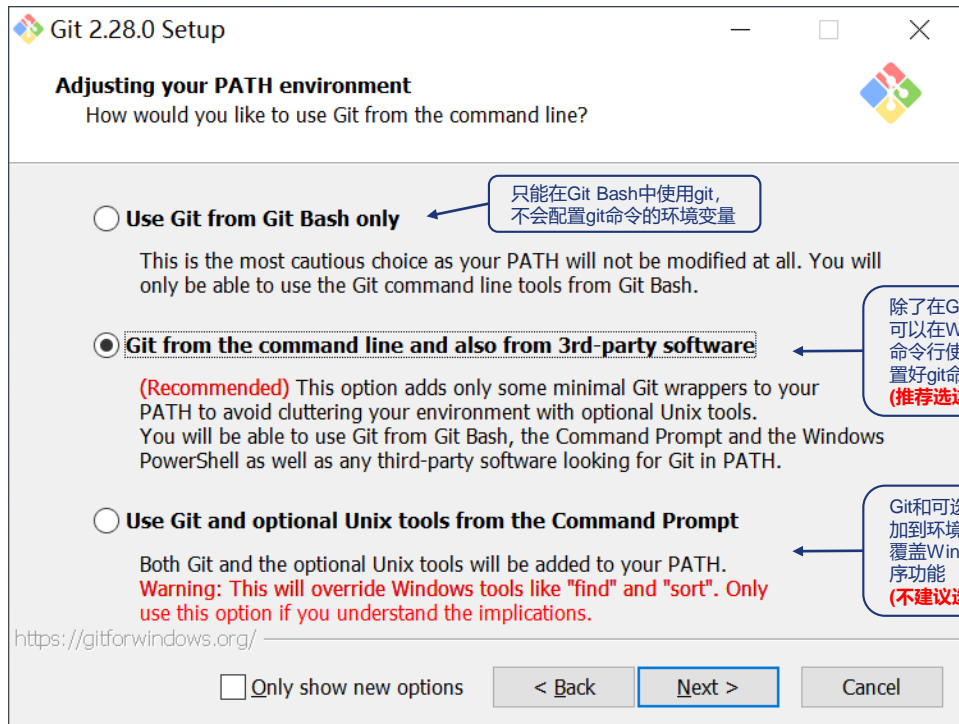


双击安装包打开安装界面



windows下安装:

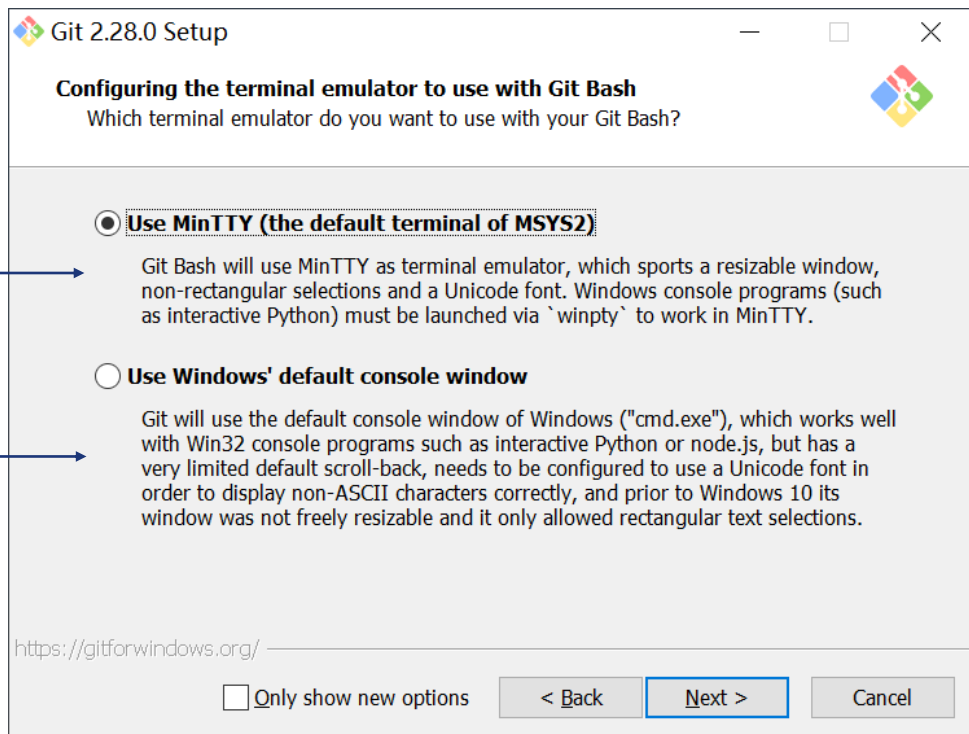
配置git的安装组件，默认
就行，继续点击Next下一
步



windows下安装:

Git使用MinTTY作为终端模拟器, Git的打开窗口可以自由调整大小
(默认选这个就行)

Git使用Windows的默认控制台窗口, Git的打开窗口不能自由调整大小



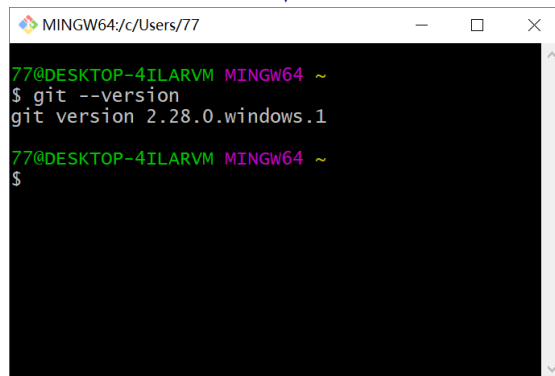
windows下安装:

最后配置Git的额外选项，选择默认就行
点击install完成安装即可

此时，桌面会有Git的快捷方式（如果你有勾选过）
或者 任意目录下右击 会有Git GUI Here, Git Bash Here的快捷方式



输入: git --version



Git常用指令

- `init` //初始化本地Git仓库
- `config` //配置用户信息
- `add` // 新增文件或文件夹
- `commit` // 提交修改
- `status` // 查看当前Working Tree的状态
- `log` // 查看日志
- `branch` // 创建分支
- `gitk` // 图形化界面
- `push` // 推送到远程仓库
- `pull` // 从远程仓库拉取更新
- `clone` // 克隆仓库到新的目录

参考资料: [Git command list](#) 【指令列表】

建Git仓库

两种场景：

1. 把已有的项目纳入Git管理

```
$ cd 项目代码所在的文件夹  
$ git init
```

2. 新建的项目直接用Git管理

```
$ cd 某个文件夹  
$ git init your_project // 会在当前路径创建项目名称同名的文件夹  
$ cd your_project
```

最小配置

配置 user 信息:

- `git config --global user.name 'yourName'` // 配置user.name
- `git config --global user.email 'yourEmail'` // 设置user.email

参数:

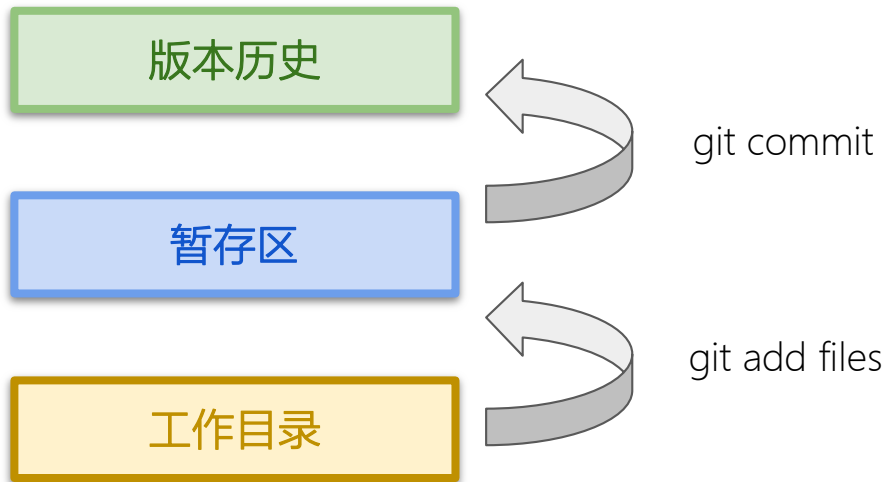
`--local` // 对某一个仓库起作用

`--global` // 对该用户的所有仓库起作用（最常用）

`--system` // 对登录系统的所有用户起作用（一般不使用）

`--list` // 显示config的配置

Git的工作原理

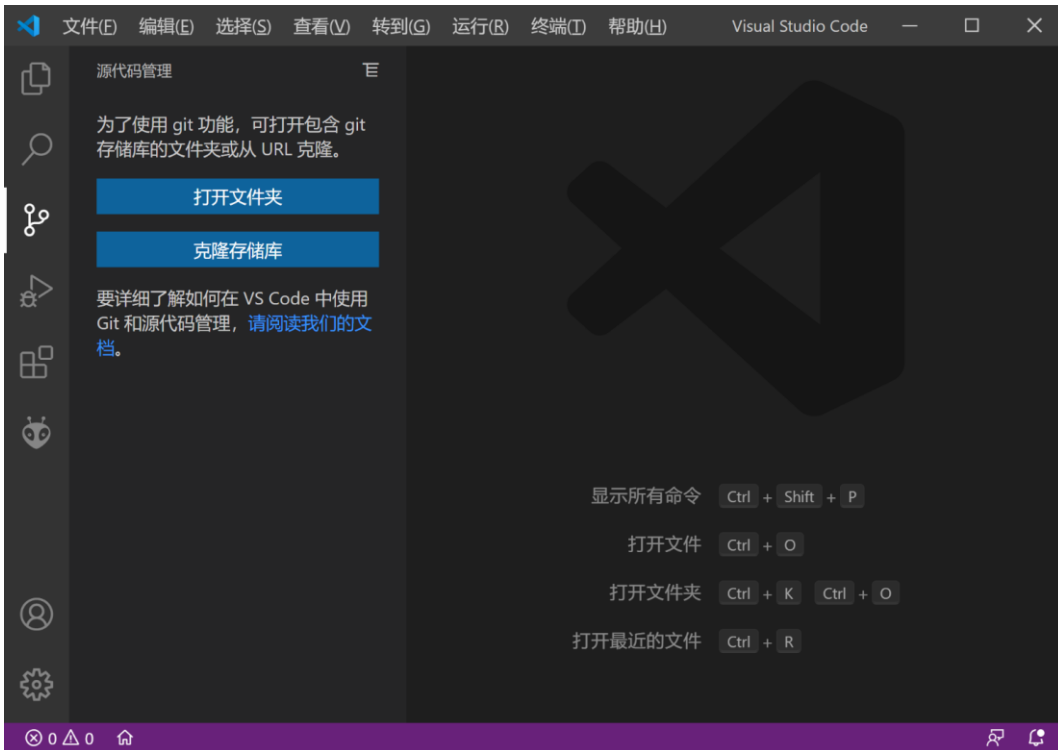


Git in VSCode

VSCode Git support 帮助文档:

https://code.visualstudio.com/docs/editor/versioncontrol#_git-support

VSCode默认已经集成了
源代码管理器(Ctrl+Shift+G)
但要使用Git, 得先安装客户端



Git in VSCode

为了使用 git 功能，可打开包含 git 存储库的文件夹或从 URL 克隆。

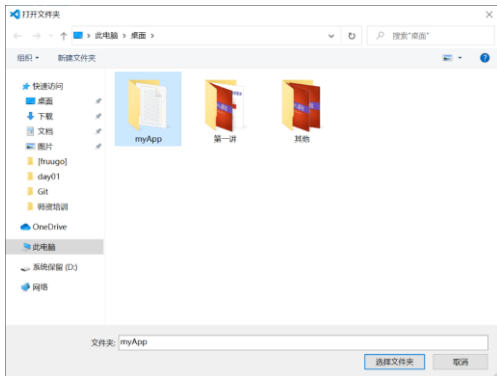
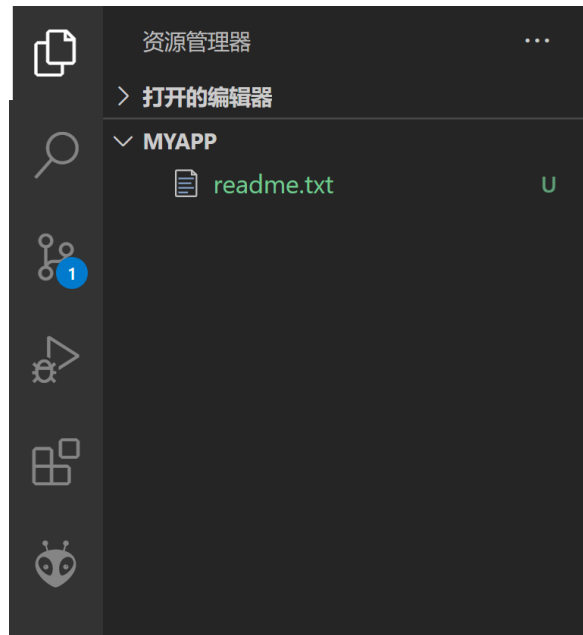
打开文件夹

克隆存储库

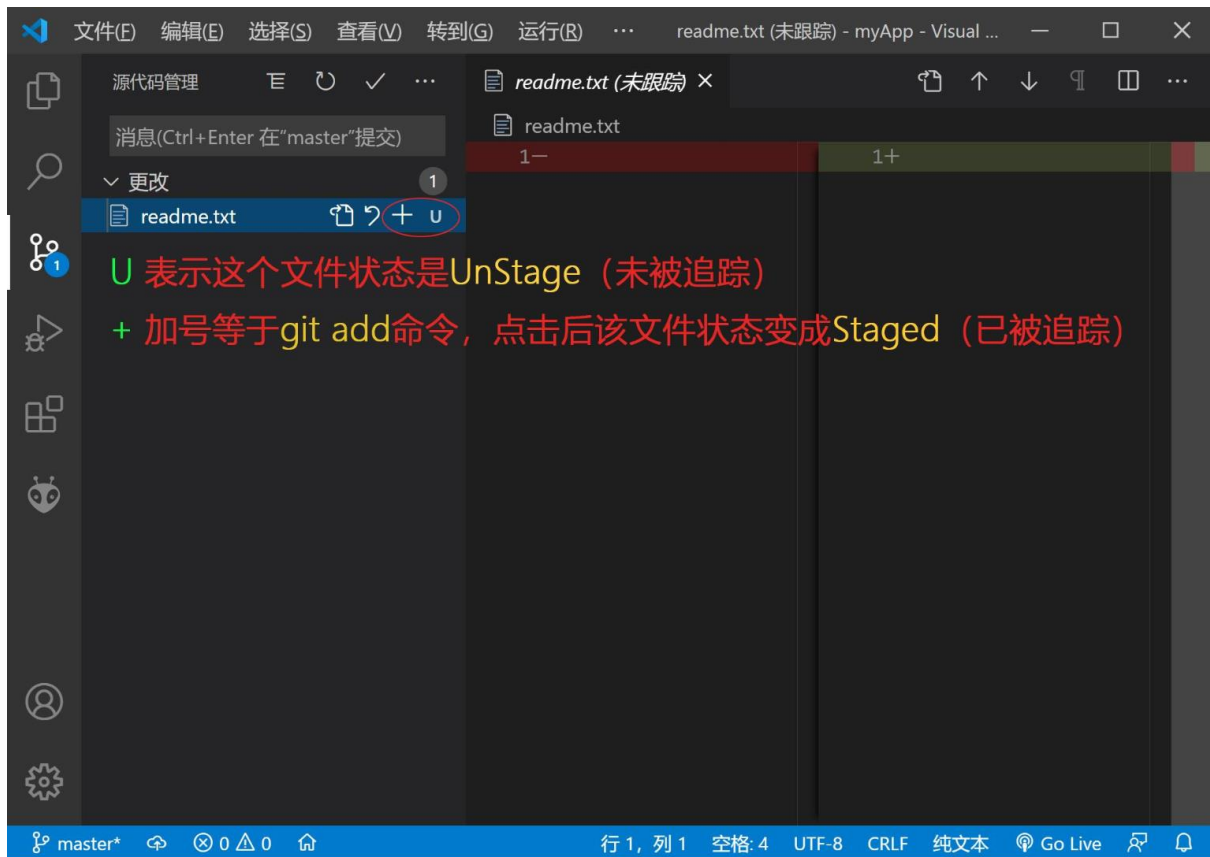
要详细了解如何在 VS Code 中使用 Git 和源代码管理，[请阅读我们的文档](#)。

点击打开文件夹

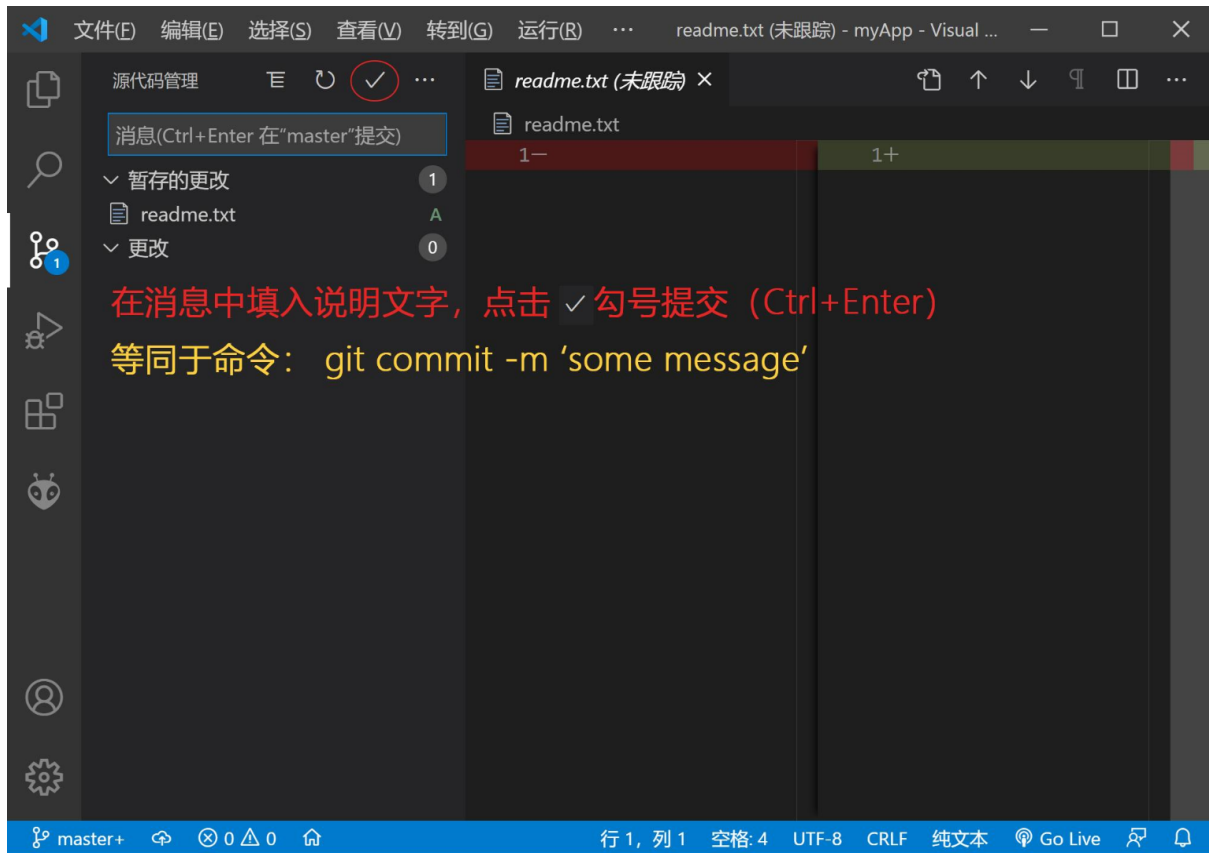
选择你想要git来管理的项目文件夹



Git in VSCode



Git in VSCode



学习参考

这些资料会带你轻松入门：

- [Hello World 【欢迎来到新世界】](#) (来自 GitHub)

在这里你可以学到 Git 的基础操作，例如：创建储存库 (Repository) 和分支 (Branch)、确认提交 (commit) 以及打开和合并拉取请求 (open/merge Pull Request)。

- [Git Git Handbook 【使用手册】](#) (来自 GitHub)

它解释了什么是版本控制系统、什么是储存库 (Repository)、GitHub 模型如何运行、Git 指令和示例等。

- [Forking Projects 【复刻项目】](#) (来自 GitHub)

当你想要对别人的代码做出贡献时，复刻 (fork，即服务端代码仓库复制) 是必不可少的步骤。这个链接将告诉你如何操作。

- [About Pull Requests 【关于拉取请求】](#) (来自 GitHub)

这是一个管理拉取请求的有用的指南：你所修改的代码需要通过拉取请求来让项目的管理者决定是否采纳。

- [Mastering issues 【处理问题】](#) (来自 GitHub)

问题区 (Issues) 就像是一个关于你这个 GitHub 项目的论坛，在这里人们可以提问题和报告错误，你可以管理更新（例如给人们分配需要解决的问题、澄清问题以及告知问题已经解决）。这篇文章会让你知道所需要的一些关于问题区的知识。

还可以看看：

- [Understanding the GitHub flow](#) 【理解 GitHub 流程】
- [Git command list](#) 【指令列表】
- [Getting Started with GitHub Pages](#) 【入门 Github 页面】（如何在 GitHub 上发布示例和网站）。
- [Learn Git branching](#) 【学习 Git 的分支结构】
- [Flight rules for Git](#) 【Git 中的飞行法则】（在Git中实现特定功能的非常实用的方法介绍纲要，包括如何在出错时纠错等）。
- [Dangit, git!](#) 【#网络和谐#，Git !】（另一个十分实用的方法介绍纲要，特别是在出错的时候进行纠正的方法）。

作业

实验题：理解Github的流程

基于你之前的课程案例项目，在Github上创建一个仓库（Repository），创建一个分支（Branch），在这个分支上做一些修改，提交（Commit）修改，拉取请求（Open a Pull Request），讨论和审查你的代码（Discuss and Review），部署测试你的分支项目（Deploy），测试无误后，把你的分支合并到主分支（Merge）。

作业要求： 用文档记录你在Github工作流程中做了什么，并且包含各步骤关键截图。

提交时间： 下周五前