目录

[### 前提条件 1](#_Toc204700936)

[2. 彻底解决：关闭/打开自动换行符转换（推荐） 4](#_Toc204700937)

[.git仓库和main分支是什么意义和区别 4](#_Toc204700938)

[.git 仓库 4](#_Toc204700939)

[main 分支 4](#_Toc204700940)

[Git和main分支区别 5](#_Toc204700941)

在 Visual Studio Code（VS Code）中使用 Git ，可以借助其内置的 Git 集成功能，以及相关的扩展插件，实现代码版本管理。以下是具体的使用方式：

# ### 前提条件

确保已经在系统中安装了 Git ，并且在命令行中能够正常执行 Git 命令。如果没有安装，可以前往 [Git 官方网站](https://git-scm.com/downloads) 下载并安装。

### 配置 Git 信息

在 VS Code 的终端（快捷键 `Ctrl + `` ，反引号） 或者系统的命令行终端中，执行以下命令配置用户名和邮箱：

```bash

git config --global user.name "你的用户名"

git config --global user.email "你的邮箱地址"

```

### 使用 VS Code 内置 Git 功能

1. \*\*打开项目文件夹\*\*：在 VS Code 中，通过 `文件` -> `打开文件夹` ，选择你要进行版本管理的项目文件夹。如果该文件夹是一个已存在的 Git 仓库（有隐藏的 `.git` 文件夹 ），VS Code 会自动识别；如果不是，你可以在终端中进入该文件夹，执行 `git init` 来初始化一个新的 Git 仓库。

2. \*\*查看文件状态\*\*：点击左侧活动栏的 “源代码管理” 图标（通常是一个带分支的图标 ），打开 Git 面板。在这里，你可以看到当前项目中文件的状态，包括未暂存的更改（红色标识 ）、已暂存的更改（绿色标识 ）。

3. \*\*暂存文件\*\*：在 Git 面板中，点击文件旁边的加号（`+` ）图标，或者右键单击文件并选择 “暂存更改”，将文件添加到暂存区。

4. \*\*提交更改\*\*：暂存完文件后，在 “消息” 输入框中输入提交说明，然后点击 “对勾” 图标，或者按下 `Ctrl + Enter` 组合键，完成提交操作。提交的内容会保存在本地 Git 仓库中。

5. \*\*推送和拉取\*\*：如果项目关联了远程仓库（如 GitHub ），在 VS Code 的终端中执行以下命令：

- \*\*关联远程仓库\*\*：`git remote add origin <远程仓库地址>` ，例如 `git remote add origin https://github.com/username/repository.git` 。

- \*\*推送更改\*\*：`git push origin main` （这里 `main` 是分支名，根据实际情况修改 ），将本地仓库的更改推送到远程仓库。

- \*\*拉取更新\*\*：`git pull origin main` ，获取远程仓库的最新更改并合并到本地。

6. \*\*分支管理\*\*：点击 Git 面板右上角的分支名称，会弹出分支管理菜单。你可以在这里创建新分支（选择 “新建分支” ）、切换分支（选择目标分支 ）、合并分支（先切换到目标分支，然后选择 “合并分支” ，再选择要合并的分支 ）等操作。

### 安装扩展插件（可选）

VS Code 插件市场有许多与 Git 相关的扩展插件，可以提供更丰富的功能和更便捷的操作体验：

- \*\*GitLens\*\*：提供强大的代码历史查看、作者信息显示、代码比较等功能，帮助你更好地理解代码的演变过程。

- \*\*Git Graph\*\*：以图形化的方式展示 Git 仓库的分支和提交历史，方便进行分支管理和提交追溯。

安装插件的方法是：点击左侧活动栏的 “扩展” 图标（通常是一个方块拼图的图标 ），在搜索框中输入插件名称，找到后点击 “安装” 按钮即可。

通过以上步骤，你就可以在 VS Code 中方便地使用 Git 进行代码版本管理，提升开发效率。

### 2. 彻底解决：关闭/打开自动换行符转换（推荐）

在 **Git Bash（或终端＿/strong> 里执行命令，关闭 Git 自动转换换行符的功能＿**

git config --global core.autocrlf false

**打开 Git 自动转换换行符的功能**

git config --global core.autocrlf true

# .git仓库和main分支是什么意义和区别

.git 仓库和 main 分支是 Git 中两个不同但又紧密相关的概念，以下是它们的意义和区别：

### .git 仓库

* **意义**
  + **存储数据**：.git 是一个隐藏的文件夹（在 Windows 系统中默认隐藏，需要开启显示隐藏文件才能看到 ，在 Linux 和 macOS 中也以隐藏形式存在），它是 Git 用来存储版本控制相关信息的地方。它包含了项目的所有历史记录，包括每次提交（commit）的快照、分支信息、标签信息、索引（暂存区的内容）等。
  + **版本控制核心**：所有的版本控制操作，比如查看历史版本、回滚变更、合并分支等，都依赖于 .git 仓库中存储的数据。可以将其看作是整个项目版本控制的 “数据库”。
* **创建方式**：当在项目目录下执行 git init 命令时，Git 就会在该目录下创建一个 .git 文件夹，从而将这个普通目录转变为一个 Git 仓库。如果是通过 git clone <远程仓库URL> 命令克隆远程仓库到本地，那么克隆下来的项目文件夹中也会包含一个 .git 仓库，它包含了远程仓库的初始数据以及与远程仓库的连接信息。

### main 分支

* **意义**
  + **主分支**：main 是 Git 中默认的主分支名称（在过去，默认主分支名称通常是 master，后来许多项目和 Git 社区逐渐将默认主分支名称改为 main，以避免 master 这个词可能带来的负面联想 ）。它通常代表项目的稳定版本，是项目发布和部署的主要分支。
  + **开发汇聚点**：在团队开发中，开发者们会在各自的功能分支上进行开发，完成功能开发和测试后，通常会将功能分支合并到 main 分支，以此来更新项目的主版本。
* **创建方式**：在使用 git init 初始化仓库时，Git 会自动创建一个 main 分支（或 master，取决于 Git 的配置和版本）。当使用 git clone 克隆远程仓库时，远程仓库的 main 分支会被同步到本地仓库，本地仓库也会有一个对应的 main 分支。

### Git和main分支区别

* **概念范畴**：.git 仓库是一个整体的存储结构，涵盖了项目版本控制的所有信息；而 main 分支只是 .git 仓库中众多分支中的一个，它只是项目开发过程中特定的一条开发线路，用来管理和组织代码提交。
* **数据内容**：.git 仓库包含了所有分支的历史记录、暂存区数据、配置信息等；而 main 分支只包含在该分支上进行的提交记录，它是项目历史记录的一个子集。
* **操作对象**：对 .git 仓库的操作通常涉及仓库的整体管理，比如设置远程仓库地址（git remote 命令 ）、备份仓库数据等；对 main 分支的操作主要围绕分支本身，比如切换到 main 分支（git checkout main ）、合并其他分支到 main 分支（git merge <其他分支名> ）等。

#### 分支main 详细介绍

在 Git 中，本地分支（如 main 分支 ）并不是像普通文件夹那样，有一个专门独立的实际路径去存储文件。

Git 存储文件和分支的机制

文件存储：你在本地仓库目录（从你的截图可知，路径是 E:\备份盘\带零文件夹\005\_计算机科学、程式、资料,硬件\005\_400\_电脑编程\1Scripts\005\_490\_运行与调试 ）下看到的所有文件，就是实际存储在磁盘上的文件。当你对这些文件进行修改、添加、删除等操作后，Git 会通过哈希算法（SHA-1 哈希 ）来记录文件的状态。

分支存储：分支本质上是一个指向提交对象的可变指针。本地分支（如 main 分支）的信息，存储在本地仓库隐藏的 .git 文件夹中。具体来说：

分支指针：在 .git/refs/heads 目录下，会有一个和分支同名的文件，例如 main 分支对应的就是 .git/refs/heads/main 文件，这个文件里记录了当前 main 分支所指向的最新提交的哈希值。

提交记录：Git 会把每次提交（git commit ）的信息存储在 .git/objects 目录下，这些提交信息包括提交的作者、时间、提交说明，以及指向父提交的指针等。通过分支指针指向的提交哈希值，Git 可以找到对应的提交记录，然后沿着提交记录的链条（因为提交记录里包含父提交指针，能形成链条追溯历史 ），获取到整个分支的历史变更。

举例说明

假设你在本地仓库创建了一个新文件 test.txt ，然后执行 git add test.txt 和 git commit -m "Add test.txt" ，此时：

test.txt 文件依然保存在你看到的本地仓库目录（比如 E:\备份盘\带零文件夹\005\_计算机科学、程式、资料,硬件\005\_400\_电脑编程\1Scripts\005\_490\_运行与调试 ）中。

Git 会在 .git/objects 目录下生成相关的对象（包括提交对象、树对象、blob 对象等 ），记录这次提交的详细信息。

main 分支对应的文件 .git/refs/heads/main 会更新，记录下这次新提交的哈希值，这样 main 分支就指向了最新的提交状态。

简单来说，文件保存在你看到的本地仓库目录中，而分支信息（指向的提交 ）存储在 .git 文件夹的相关目录里 ，通过这种方式，Git 可以高效地管理版本和分支。

# 命令

git pull my---first main --allow-unrelated-histories 允许合并 “无关历史"

git branch –vv 本地分支关联的远程分支

git push -u origin main 重新推送 / 关联: