

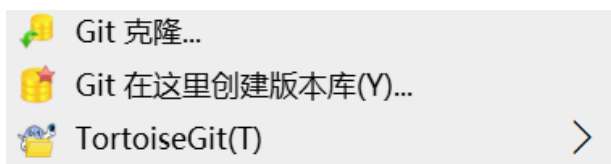
# 面向 njm\_op 项目 git 使用说明

## 【确保事项】

1) 请确保已安装 git.exe, 在命令行输入 git version:

```
C:\Users\lzl>git version  
git version 2.33.0.windows.2
```

2) 推荐安装图形化 git 工具, 这里与后文都以 TortoiseGit 小乌龟为例, 如果安装且配置成功, 右键任意位置可以出现下面的选项:

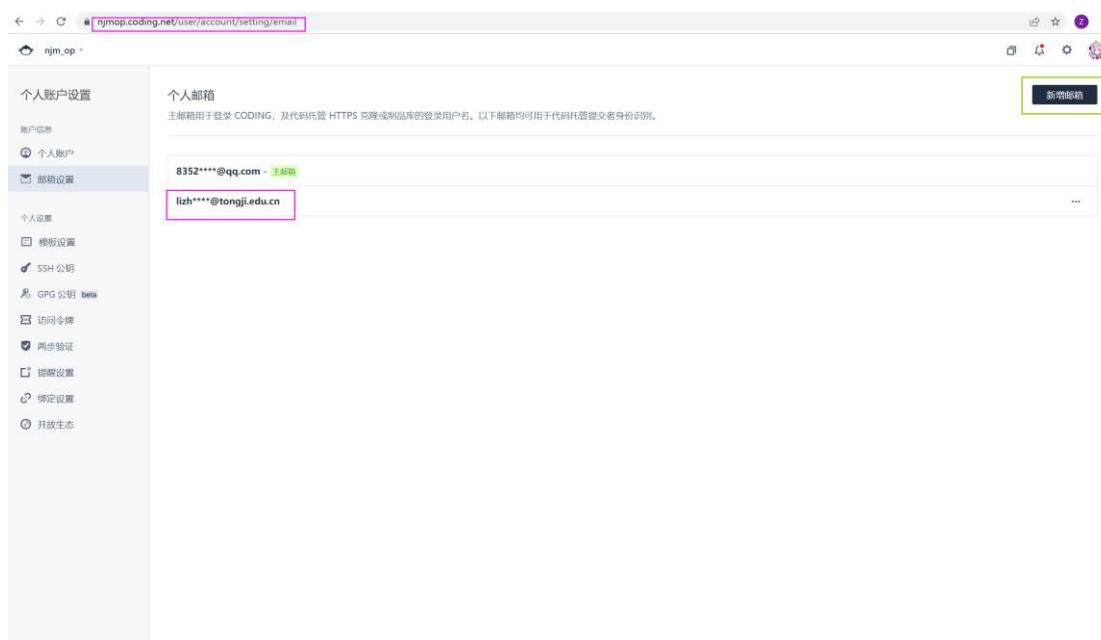


3) 务必确保自己在 git.exe 上注册的邮箱和在 coding 平台用于身份认证的邮箱是对应的。

在命令行可以查看自己的 git.exe 注册的邮箱:

```
C:\Users\lzl>git config user.email  
lizhilin@tongji.edu.cn
```

在 coding 平台可以查看自己在 njm\_op 上的身份识别邮箱列表:

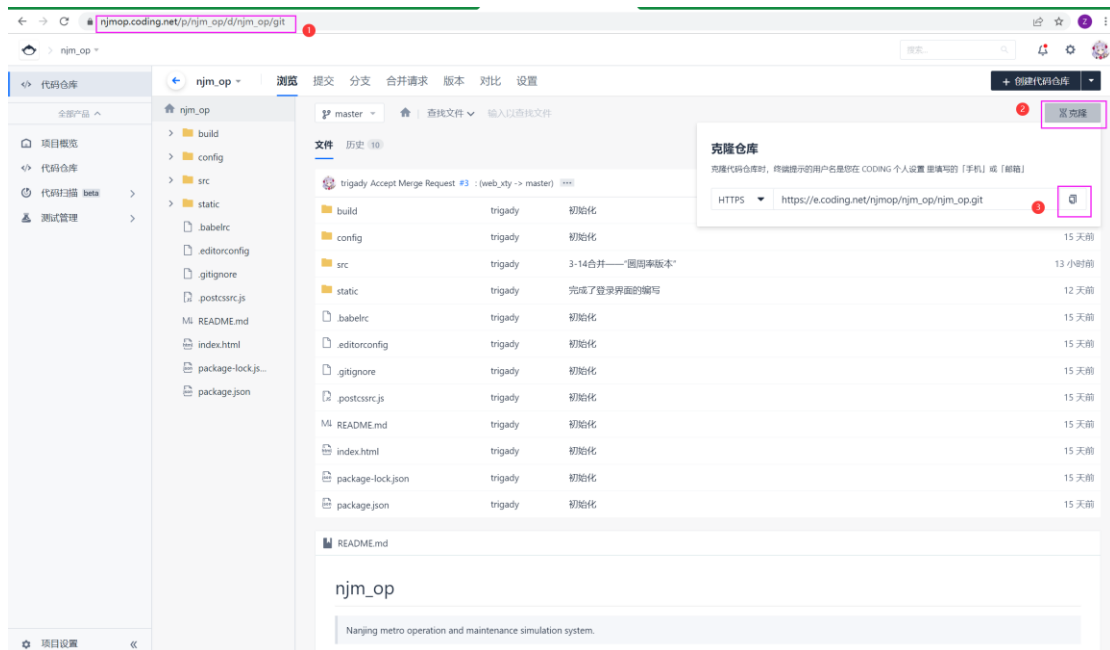


一定要确保自己在 git.exe 上注册的邮箱在这个邮箱列表中，不然之后进行 git 权限管理的时候你就无法提交代码了。

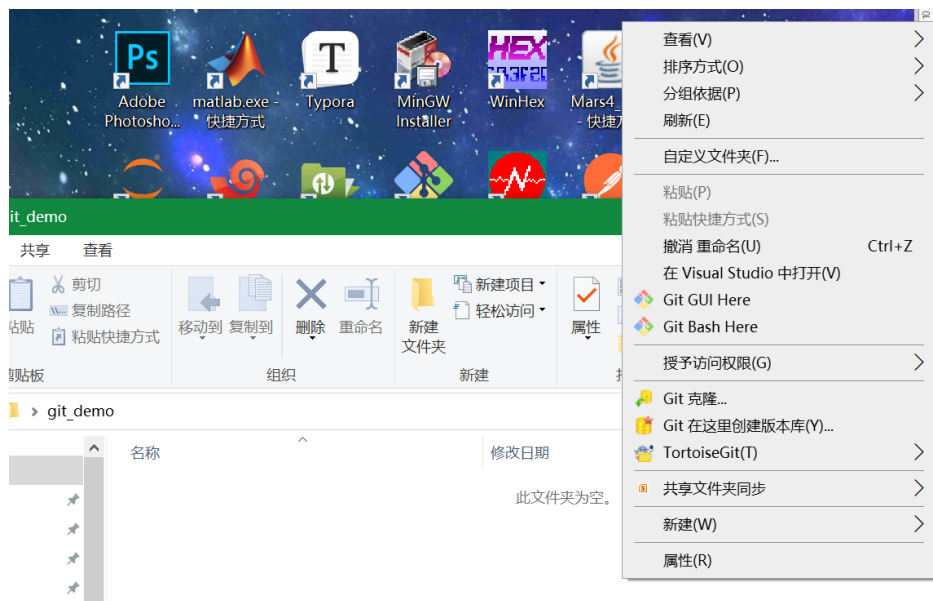
4) 可能需要删除你现在的工程文件，因为 zip 下载的工程不包含.git隐藏文件夹，没法加入版本管理。

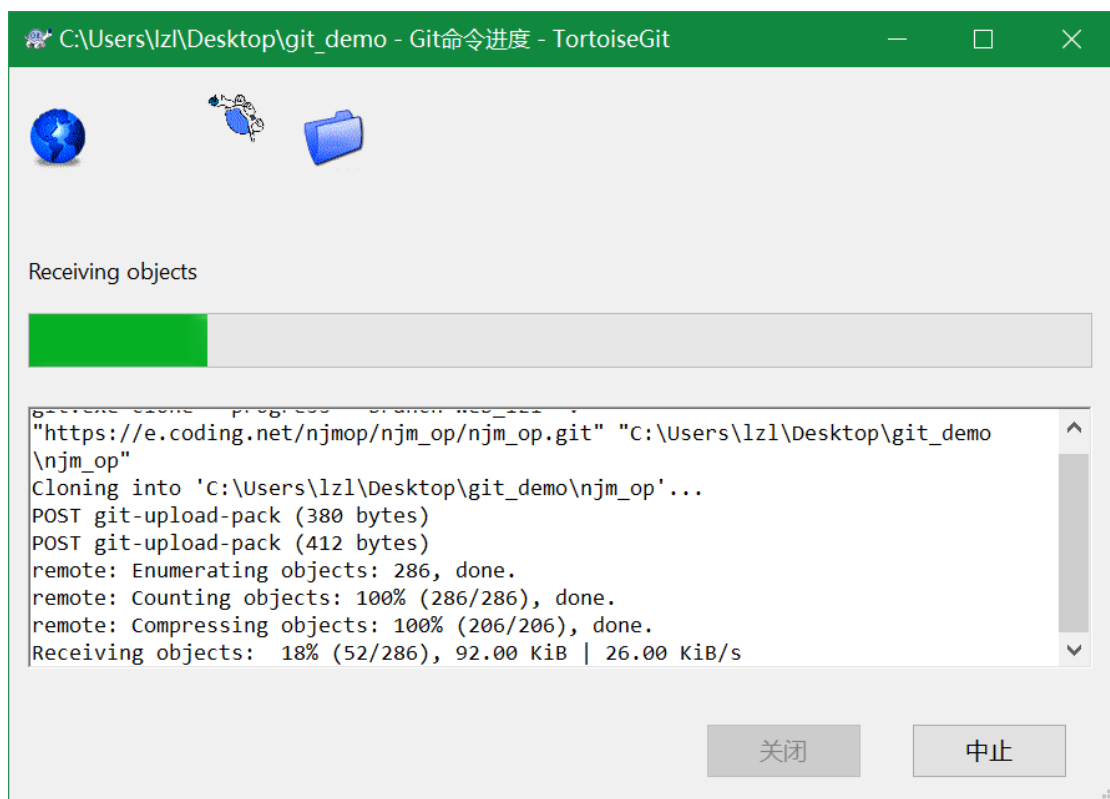
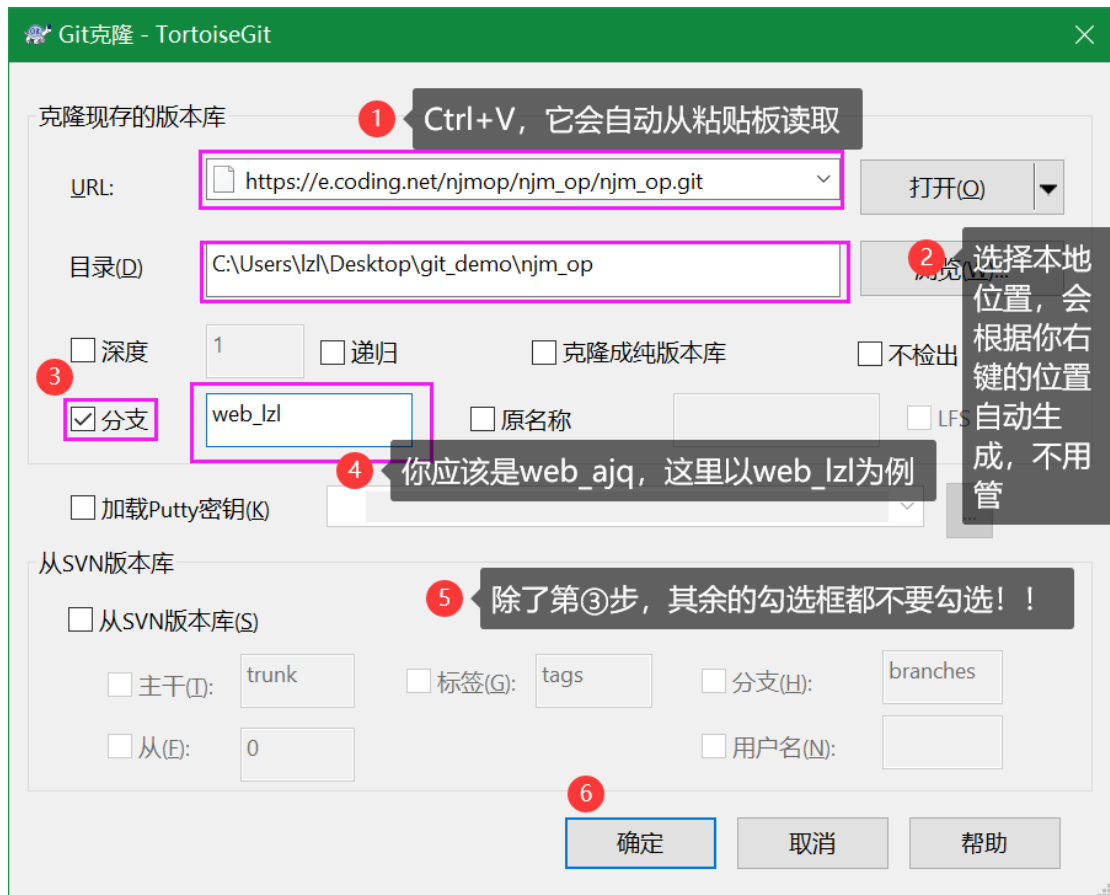
## 【首次拉取工程】

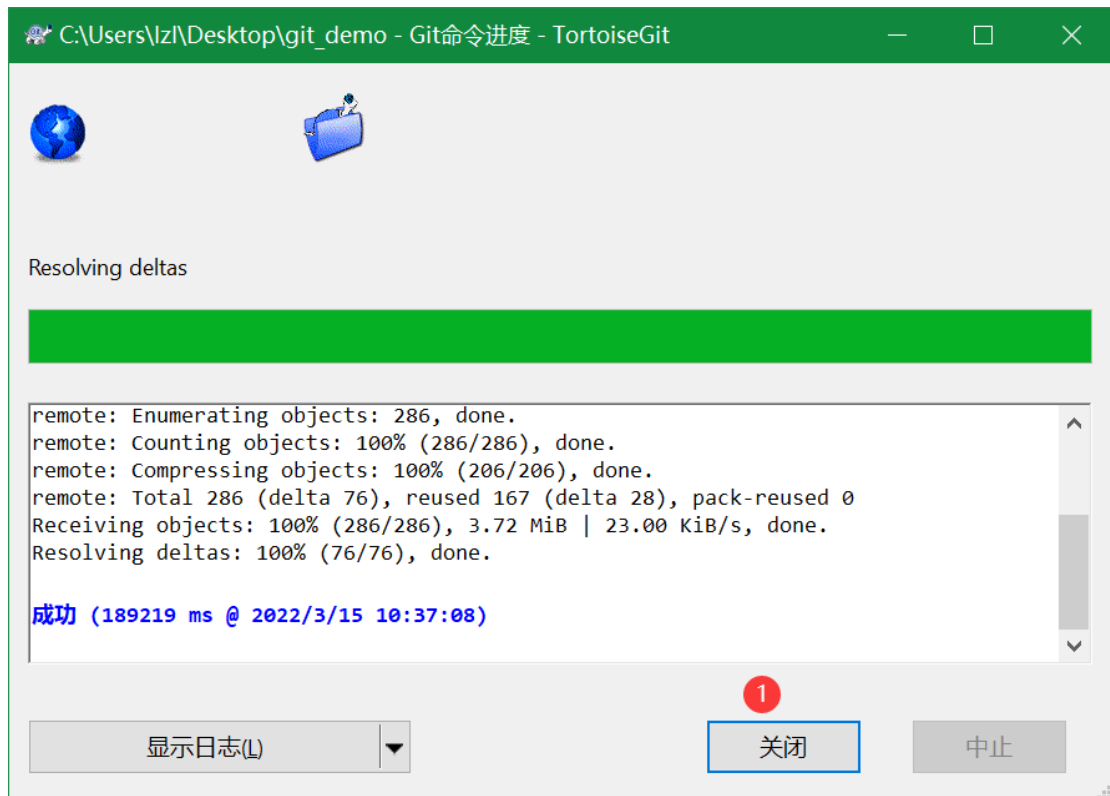
1) 进入 coding 工程，复制 git 仓库链接：



2) 在合适的位置建立文件夹，右键选择 Git 克隆...：







在原目录可以看见刚才拉取下来的版本：



然后就是之前那一套，在这个文件夹下开发就好。

P. S.

小乌龟管理的 git，文件（夹）左下角的绿标表示这个文件（夹）是在版本控制范围内的，且其相比于上个拉取或是推送的版本没有进

行过更改。

这也就是为什么 npm install 之后，生成的 node\_modules 没有绿标：

|              |                 |     |
|--------------|-----------------|-----|
| config       | 2022/2/28 16:28 | 文件夹 |
| node_modules | 2022/3/1 17:44  | 文件夹 |
| src          | 2022/2/28 16:28 | 文件夹 |
| static       | 2022/2/28 16:28 | 文件夹 |

### 【对已在版本控制的文件进行修改】

1) 譬如我对 njm\_op\src\views\title\_bar\index\_content\index\_content.vue 进行修改：

```
<div style="display: inline-block; margin-left: 1vw; cursor: pointer;">
  
  <div style="display: inline-block; vertical-align: middle; font-size: 0.9vw; padding-left: 0.2vw">显示过滤</div>
</div>
<div style="display: inline-block; margin-left: 1.5vw; cursor: pointer;">
  
  <div style="display: inline-block; vertical-align: middle; font-size: 0.9vw; padding-left: 0.4vw">清除过滤</div>
</div>
<div style="display: inline-block; margin-left: 1.5vw; cursor: pointer;">
  
  <div style="display: inline-block; vertical-align: middle; font-size: 0.9vw; padding-left: 0.3vw">刷新123</div>
</div>
```






```
<div style="margin-bottom: 2vw; margin-top: 0.6vw; text-align: left">
  <div style="display: inline-block; margin-left: 1vw; cursor: pointer;">
    
    <div style="display: inline-block; vertical-align: middle; font-size: 0.9vw; padding-left: 0.2vw">显示过滤123</div>
  </div>
  <div style="display: inline-block; margin-left: 1.5vw; cursor: pointer;">
    
    <div style="display: inline-block; vertical-align: middle; font-size: 0.9vw; padding-left: 0.4vw">清除过滤123</div>
  </div>
  <div style="display: inline-block; margin-left: 1.5vw; cursor: pointer;">
    
    <div style="display: inline-block; vertical-align: middle; font-size: 0.9vw; padding-left: 0.3vw">刷新12377897897</div>
  </div>
</div>
```

保存文件后，就会发现该文件和该文件的所有母目录都变成了红标，因为此时其在版本控制范围内，但相比于上一个推送或是拉取的版本有变化，而这个变化是“可丢失”的。

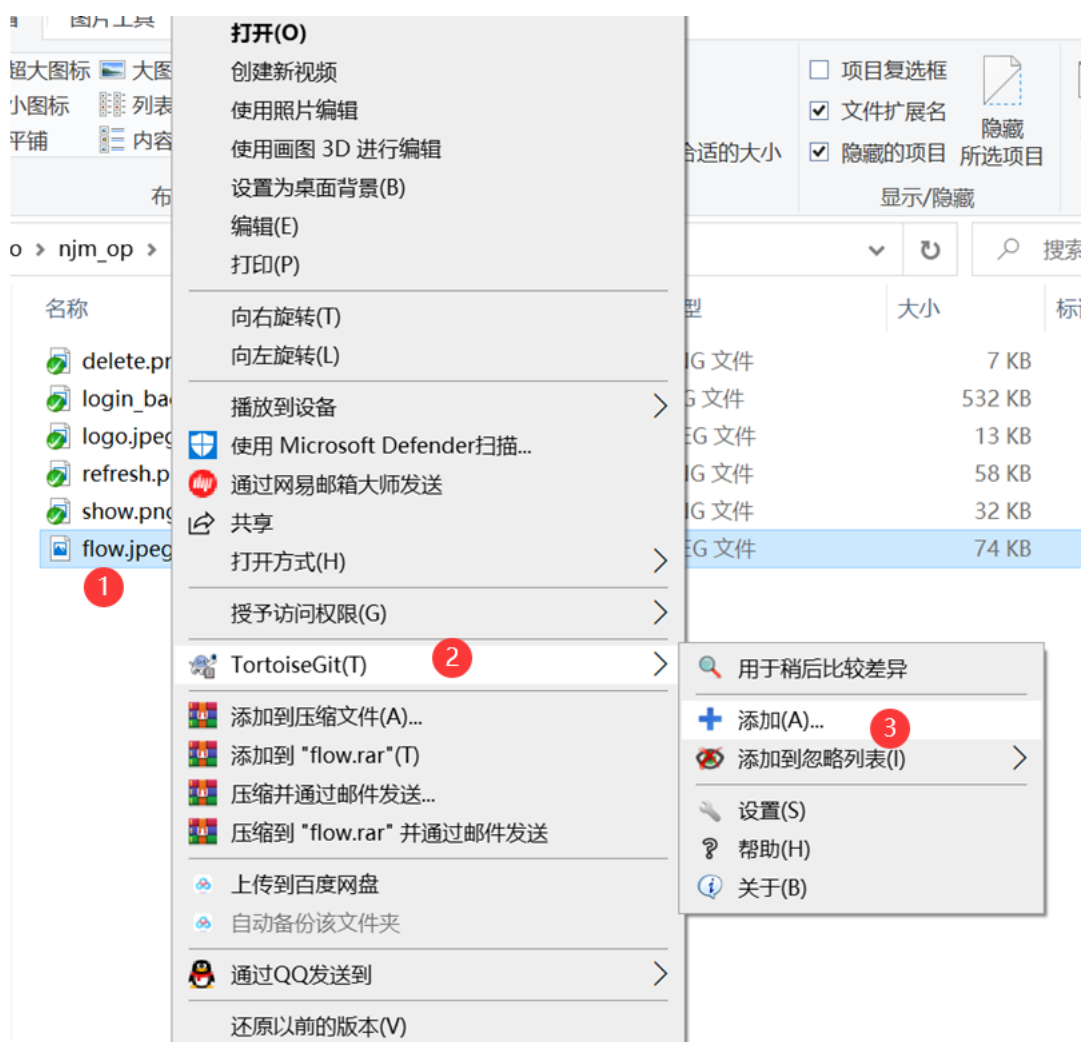
|                     |                 |            |
|---------------------|-----------------|------------|
| index_content.vue   | 2022/3/15 10:52 | router     |
| hitch_management    | static          | views      |
| material_management | App.vue         | Global.vue |
| team_management     | .....           |            |
| title_bar           |                 |            |

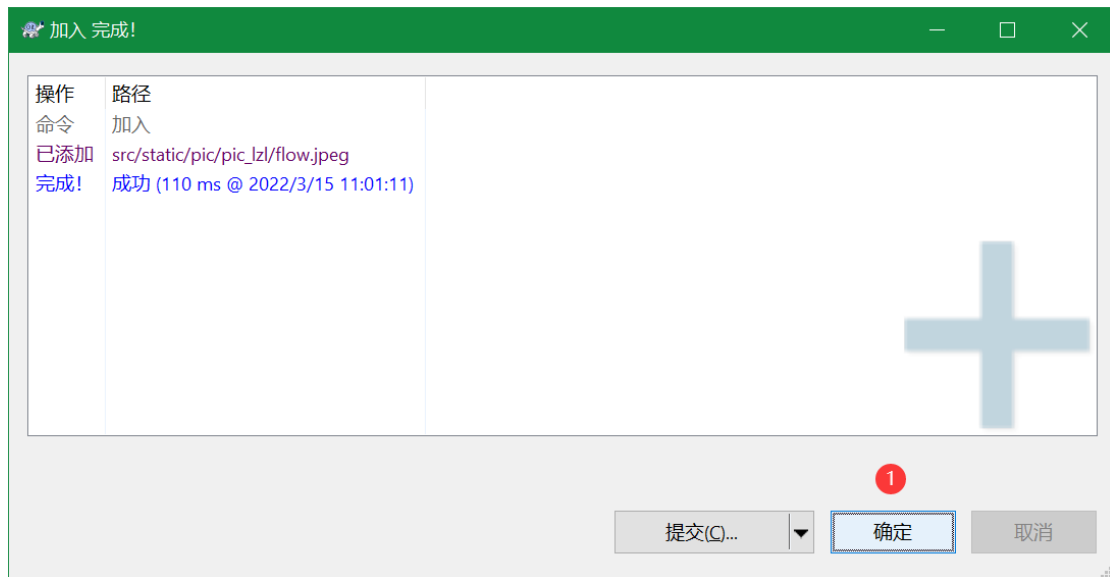
## 【对不在版本控制的文件添加控制】

1) 譬如我在 C:\Users\lzl\Desktop\git\_demo\njm\_op\src\static\pic\pic\_lzl 目录下新增一个 flow.jpeg:

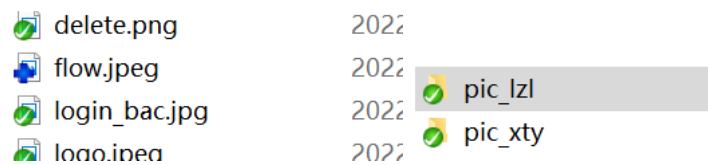
| njm_op > src > static > pic > pic_lzl   |                 |         |        |   |
|---|-----------------|---------|--------|---|
| 名称  | 日期              | 类型      | 大小     | 标 |
|  delete.png    | 2022/3/15 10:37 | PNG 文件  | 7 KB   |   |
|  login_bac.jpg | 2022/3/15 10:37 | JPG 文件  | 532 KB |   |
|  logo.jpeg     | 2022/3/15 10:37 | JPEG 文件 | 13 KB  |   |
|  refresh.png   | 2022/3/15 10:37 | PNG 文件  | 58 KB  |   |
|  show.png      | 2022/3/15 10:37 | PNG 文件  | 32 KB  |   |
|  flow.jpeg     | 2022/3/15 10:49 | JPEG 文件 | 74 KB  |   |

可以看到此时 flow.jpeg 是没有绿标的，因为其不在版本控制范围内，则：





此时发现，flow.jpeg 变成了蓝色下标，蓝色下标表示新加入版本管理的文件，但上一级依然是绿色下标：



说明：新建 vue、js 等各种文件操作同上。

禁止将 node\_modules 文件夹加入版本管理!!!

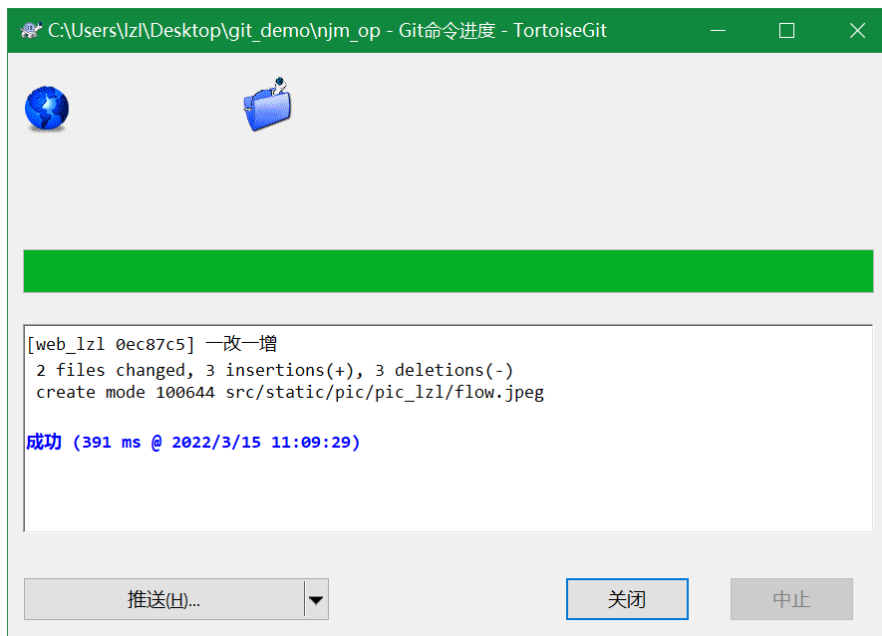
禁止将 node\_modules 文件夹加入版本管理!!!

禁止将 node\_modules 文件夹加入版本管理!!!

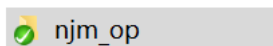
## 【推送更改】

1) 退回到主目录，右键最上层的 njm\_op，选择 Git 提交：



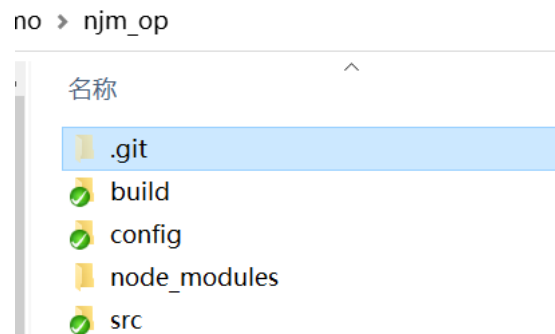


发现此时文件夹不再是红色下标:

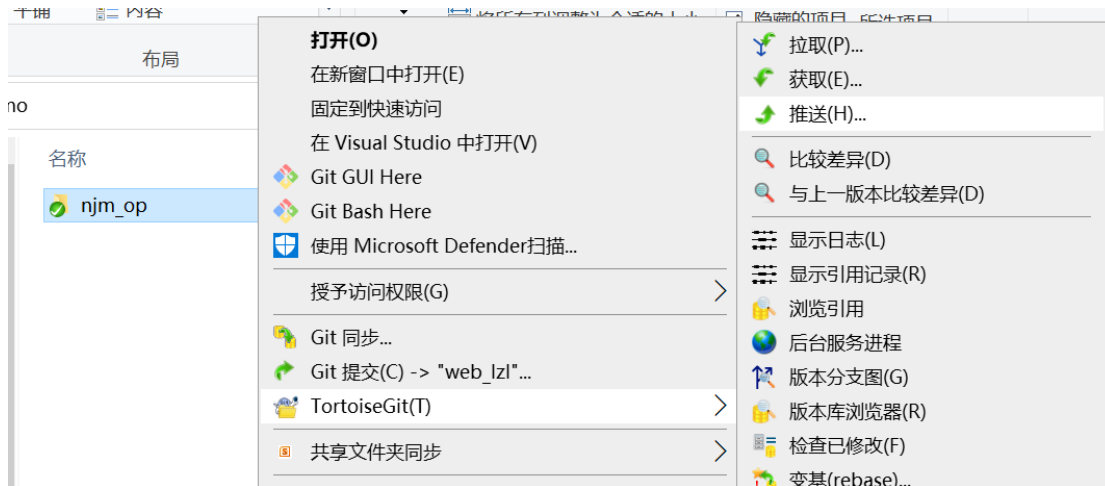


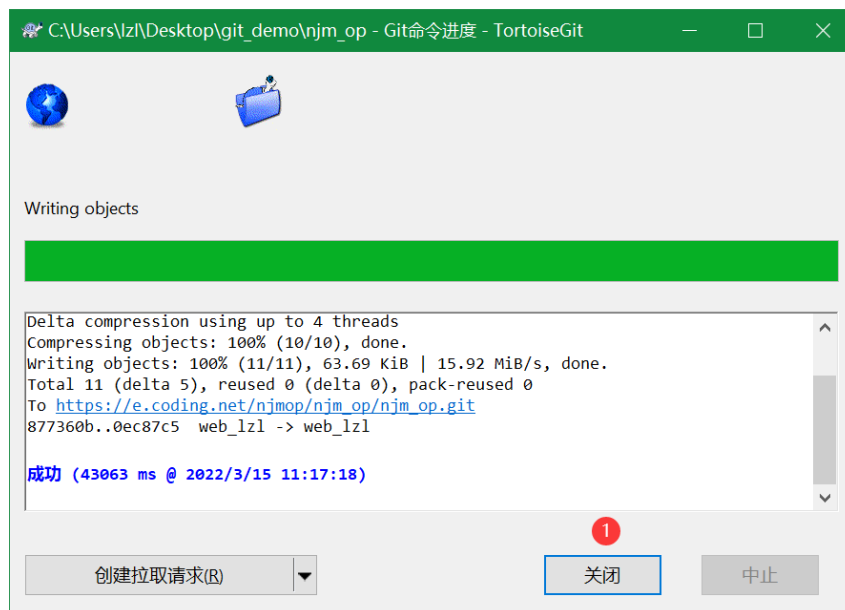
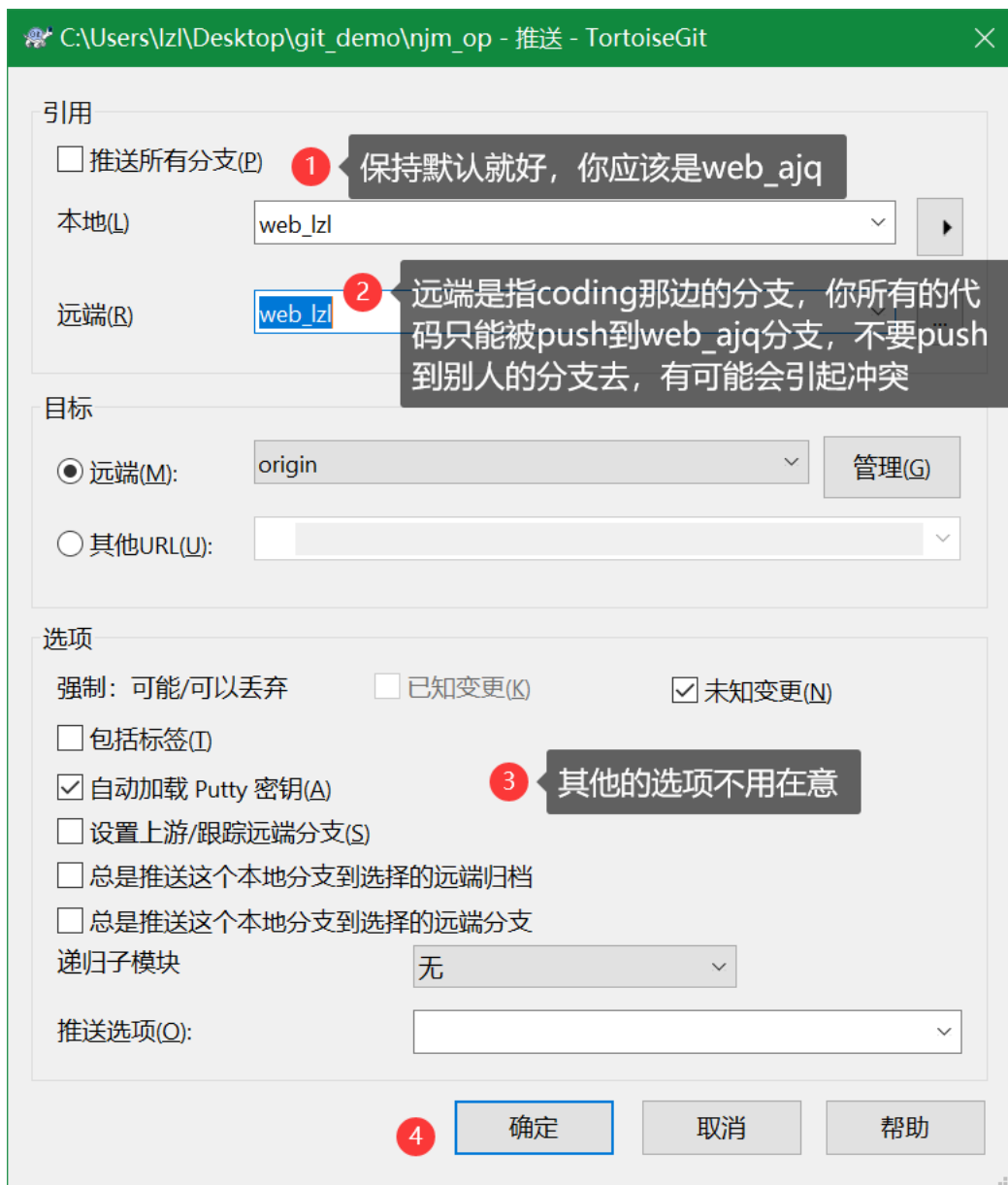


此时我们已经将修改的内容同步到了本地的仓库，所谓本地仓库，就是这个.git 隐藏文件夹中进行的仓库版本管理文件夹：

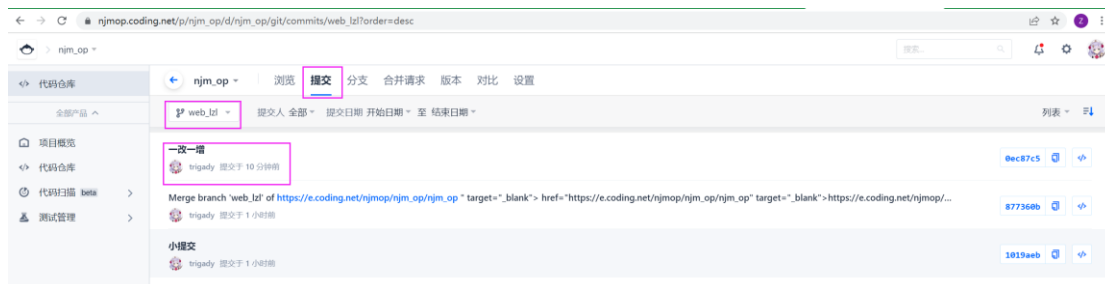


2) 回到主目录，右键最上层的 njm\_op，选择推送：

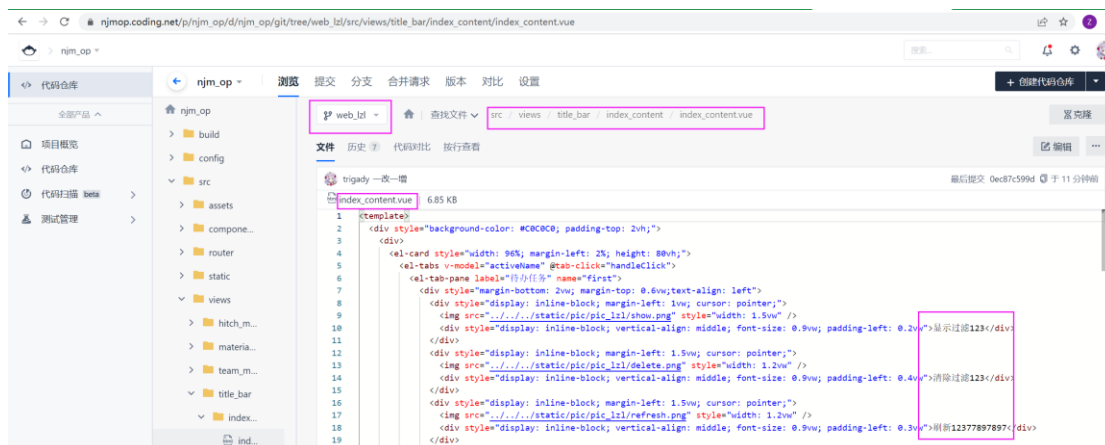




截止到此，才算完成了完整的 push 推送到云端的流程。打开 coding 平台，应该可以看到刚才的提交日志：



打开代码仓库，切换到自己的分支，打开刚才修改的文件，应当也能看见修改后的内容：



总的来说，push 推送分为两步：

- ① 把修改内容作为一个小版本推送到本地 git 仓库
- ② 从本地 git 仓库推送小版本到 coding 云端

## 【拉取更新】

除了在创建工程时需要拉取，如果我合并了大版本，推送到你们各自的分支之后，需要你们在新的版本上再开始开发时，这时候就需要你重新拉取 pull 更新。（比如 3 月 14 日的圆周率大版本这样的）

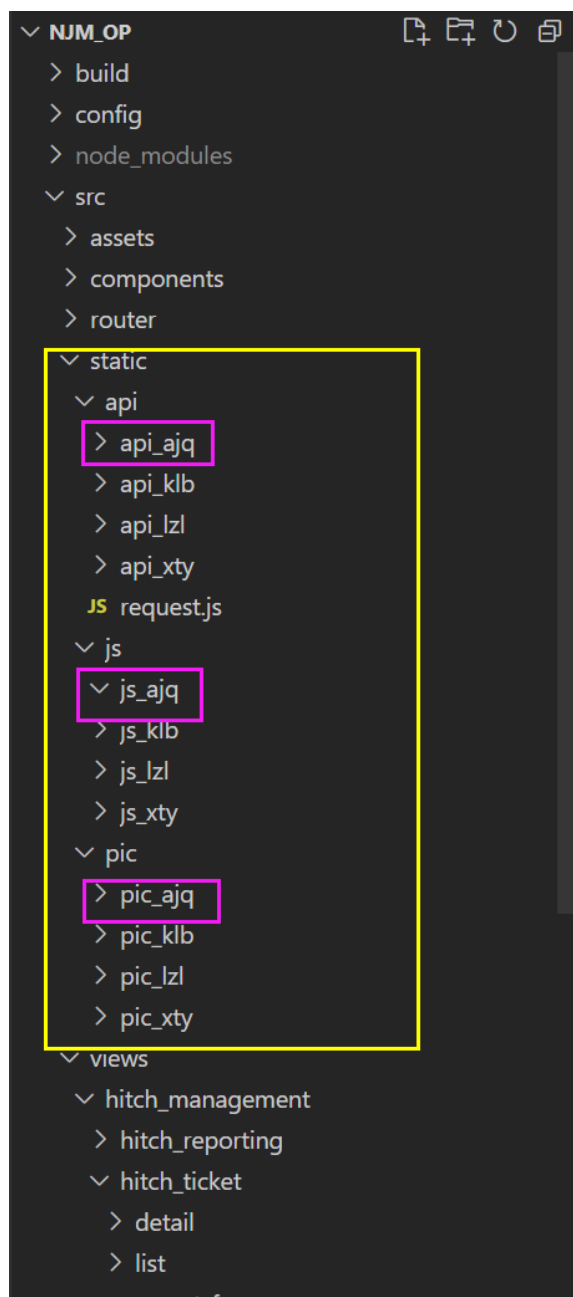
1) 退回到主目录，右键最上层的 njm\_op，选择拉取：



### 【其余情况说明】

因为 git 不支持空文件夹的存在, 所以你下载到的 static/pic 文件夹下是不会有 pic\_ajq 文件夹的, 但是我们还是希望大家的图片、js 等文件能在各自的文件夹中管理。

所以在你导入第一张图片或是第一个 js 文件前, 请先按照下图建立同名同结构的文件夹:



### 【注意事项提醒】

1) 所有代码只应被 push 到自己的分支上，禁止向别人的分支 push 代码，更禁止直接 push 到 master 分支去，每次 push 的时候烦请注意。pull 别人分支和 master 分支的代码是允许的。

2) 每次 push 代码后，可以向我说一下这次 push 大概更新了什么内容，最好能在每次鹏凯开会当天的下午四点前完成最后版本的 push，按照郭老师的意思，可能之后的工作量介绍需要你自己来进行，直接在自己的分支介绍就行。

3) 禁止将 node\_modules 加入版本管理。（逃