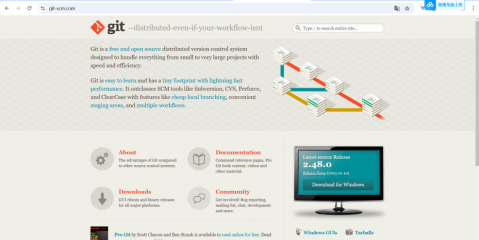
# Git使用

**Git是一种分布式版本控制系统:**不同于集中式版本控制,Git作为分布式版本控制系统不会把所有鸡蛋放在一个篮子里,所以即使有一个仓库出现了问题,也不会出现所有人都无法开发的问题。

### Git的安装

使用git前先要去官网下载一下git。傻瓜式的安装,没啥好说的。

### Git的初始化配置

**命令行**

**其实就是在终端里输入命令行对git进行控制（全局配置）**



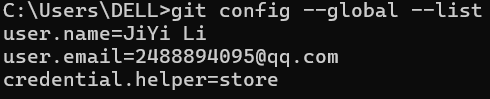
配置用户名



配置密码



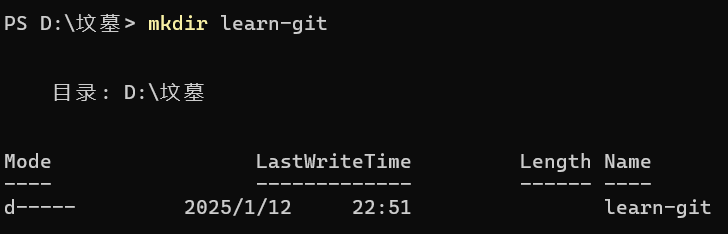
保存账号密码不用每次都配置了。



查看之前的配置,OK好用。

### Git仓库创建

#### 1.本地创建(git init 名字)

先创建一个空目录

跳转

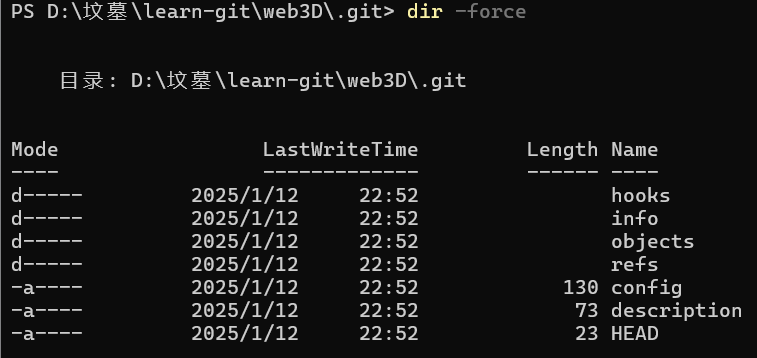
创建仓库



跳转到仓库



查看仓库

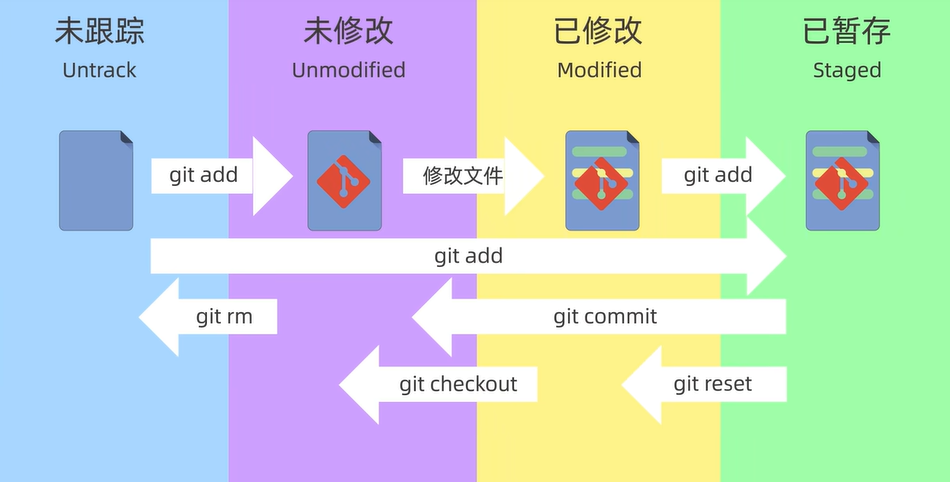


##### 2.远程克隆（git clone 地址）

### git的工作区域和工作状态

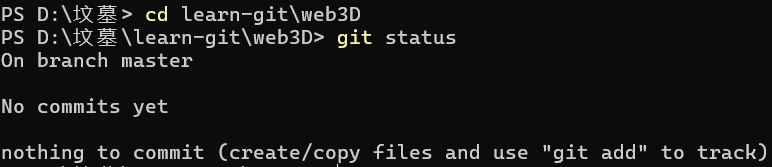


Git的工作区域分为工作区,暂存区,本地仓库。Git在工作区写代码,放到暂存区,最后一起提交到本地仓库。



文件的状态对应的就是未跟踪,未修改,已修改，已暂存。

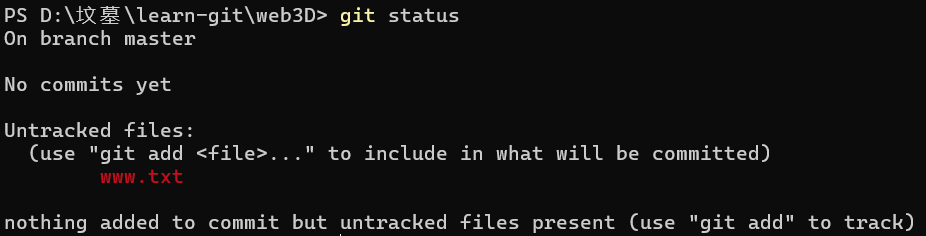
### 添加和提交文件



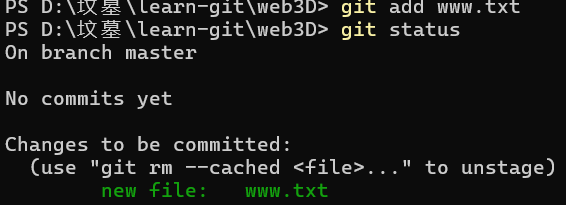
git status可以查看仓库的状态。



我们在和.git同级的目录里创建文件。

再用**git status**来看就是未跟踪状态。

用**git add**这个命令可以把文件放到暂存区。

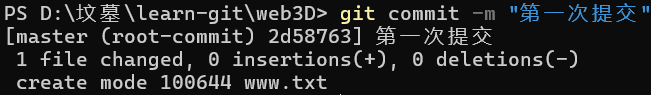


**git add**能够使用通配符，可以一次添加多个文件。



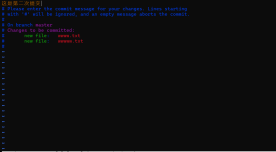
这个命令可以把所有文件都添加到暂存区里。



**git commit**把暂存区的东西放进仓库内部。

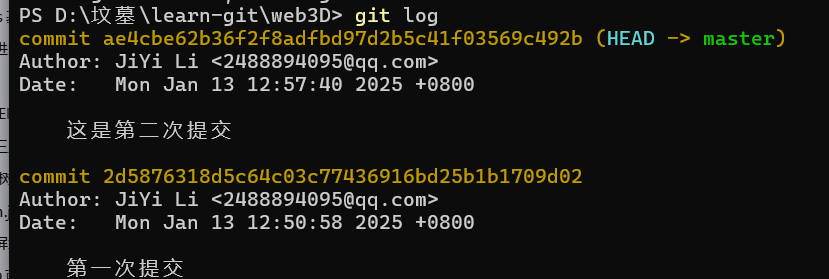
要加-m。不然会进入一个交互页面。



按i编辑提交。之后ESC退出编辑模式,再输入:wq命令退出。

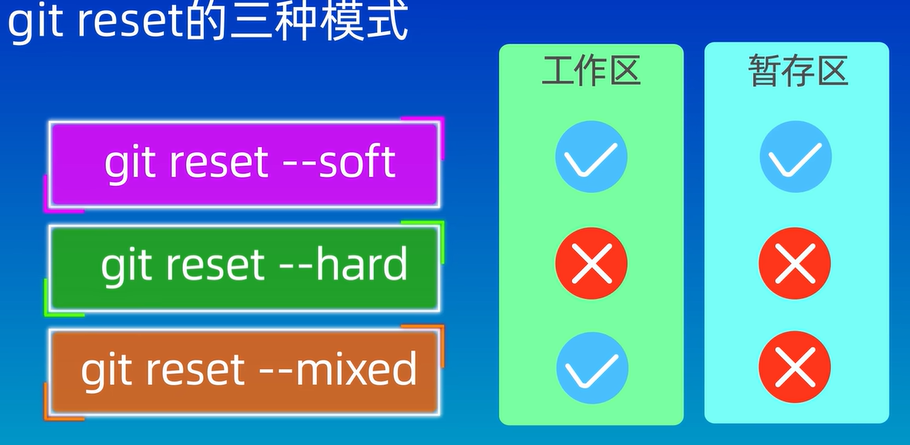
现在这个状态就是都提交了。

可以用**git log**这个命令查看提交记录。



非常的好用。

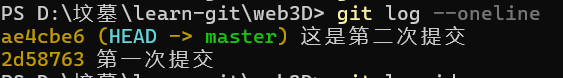
### git版本回退



进行多次没用的提交后可以回退版本用**git reset**命令。

他有三种形式：

1. soft是工作区暂存区都保留。
2. Hard是工作区暂存区都不不保留。
3. Mixed是工作区保留,暂存区不保留。

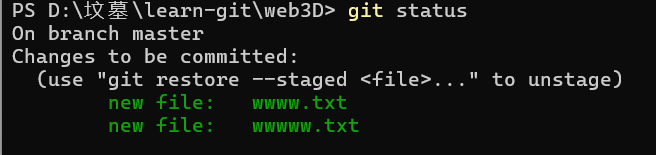


这里我们有两次提交，第二次提交错了的情况下我们要回退到第一次就用**git reset**

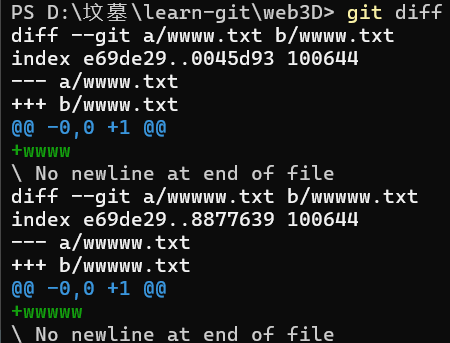


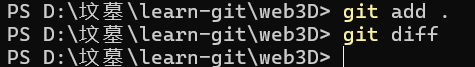
版本回退之后我们就可以看见。



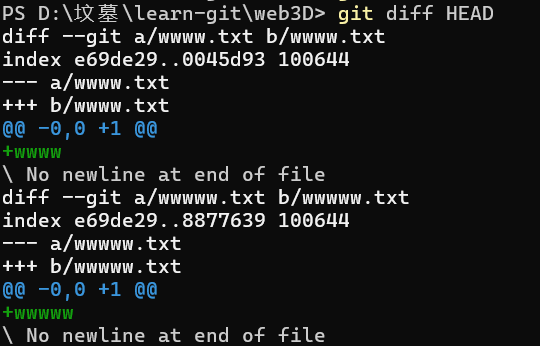
第二次提交被删除了,很好这就相当于第二次没提交嘿嘿。我们用的soft所以工作区和暂存区都给我们保留了。

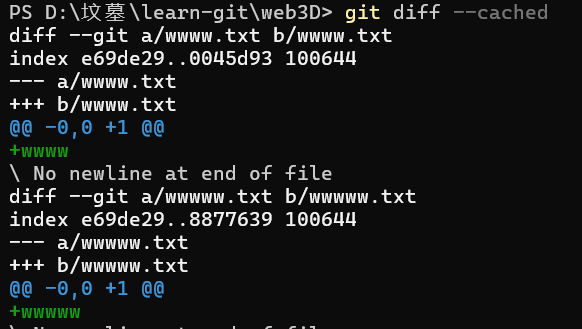
### 查看差异git diff

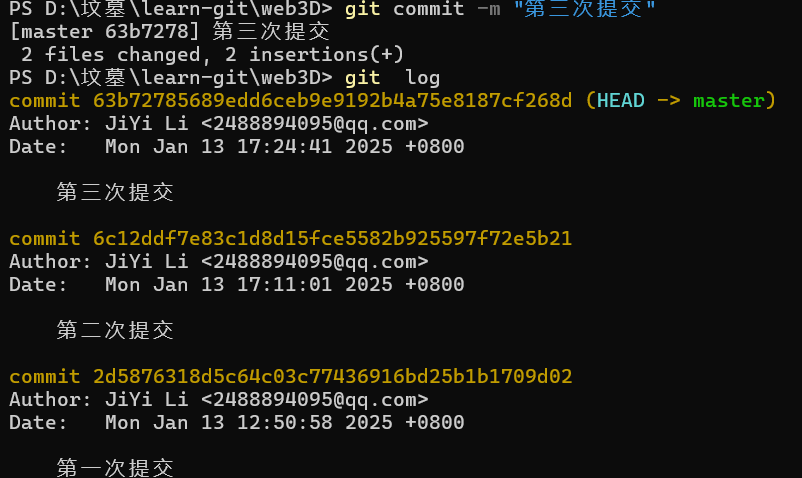
默认是比较工作区和暂存区之间的差异,我们工作区改了没去暂存区就会有差异。

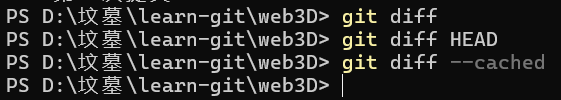


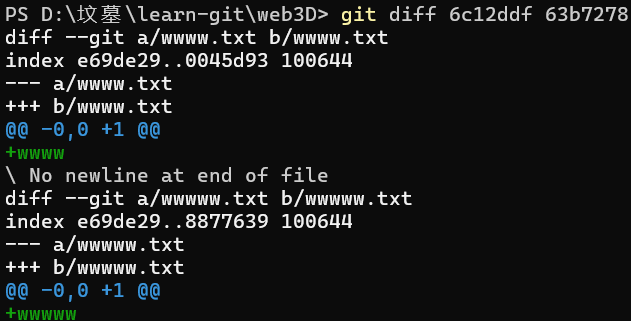
工作区内容给了暂存区就没区别了。git diff不会有任何东西。

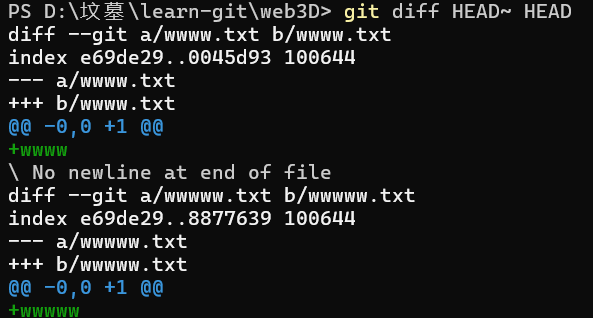
后面加HEAD就是比较工作区和版本库之间的差异。

比较暂存区和版本库直接的差异。

提交之后,工作区,暂存区和仓库没区别了。



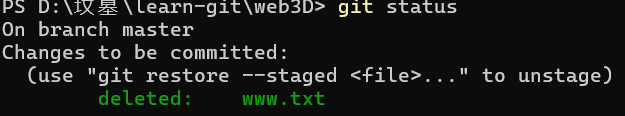
比较两个版本之间的差异。

比较当前版本和上一个版本的差异。

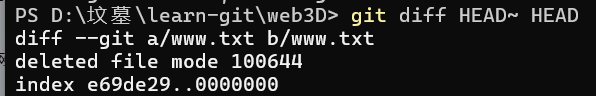
### 删除文件(git rm 文件名)

1. 工作区里直接把文件删除后,再保存到git的暂存区里。
2. 

用git rm命令在工作区和暂存区同时删掉文件。



提交一下就在仓库里删掉了。

**查看仓库文件**


www删了只剩这俩了。

### 忽略文件(.gitignore文件)

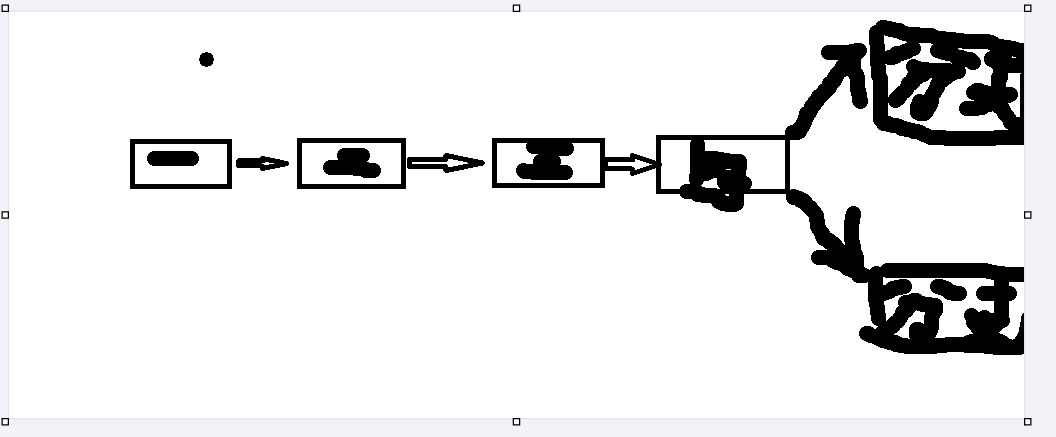
理解为一个叫做.gitignore的文件夹,里面存了所有需要忽略的文件。

### 分支

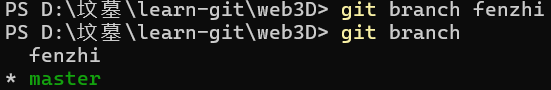
#### 分支介绍



这是四次提交记录,现在HEAD指向的是master。分支的意思就是在这四次提交记录的基础上分出一条。通常是协作开发和修改bug的时候用的。示意图大概是下面这样:

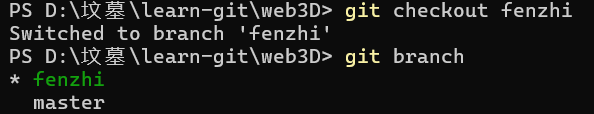
意思就是在分支里继续上面的代码进行修改,同时不影响master这个主版本。

#### 创建分支

可以看到,我们用**git branch分支名**这个命令创建了一个分支。

#### 查看分支与切换分支

用**git branch**可以查看所有分支。可以用**git checkout分支名**这个命令切换分支。



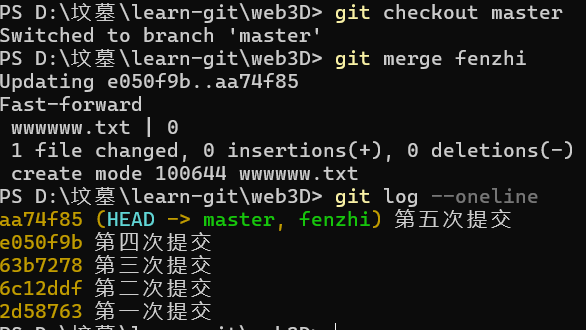
我们再进行第五次提交看看结果。



第五次提交也是完成了。可以看到我们HEAD指向了fenzhi。我们改的都是分支里的工作区和暂存区。不会对master这个主干有任何影响。

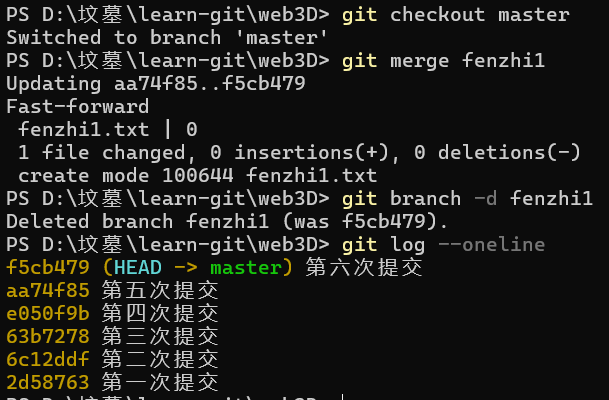
#### 分支合并，删除被合并的分支

现在创建两个分支,我们先把fenzhi也就是第五次提交这个分支合并到master主干上。

先切到主干分支,然后用**git merge 分支名**这个命令进行一个分支的合并。

合并后删除我们第五次的分支。

这样我们就完成了一个分支的合并。这个是直接合上去的。

他的合并是按提交的时间顺序来的并不是按照合并的顺序来的。所以不是所有的都是Fast-forward。

#### 合并冲突

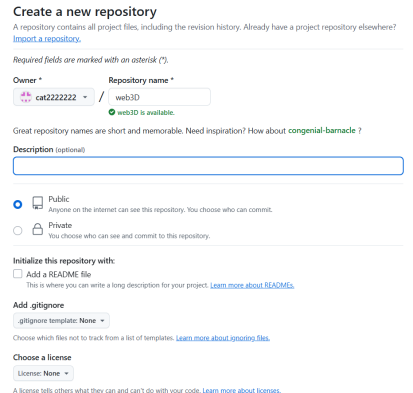
分支合并时有人他就是手贱,把不需要在分支里改的东西在分支里改了,两个分支都改了,导致和并的时候git不知道合哪个。**这种情况就得手动改掉之后再提交一下了。**

### 远程仓库

#### 创建远程仓库并关联本地

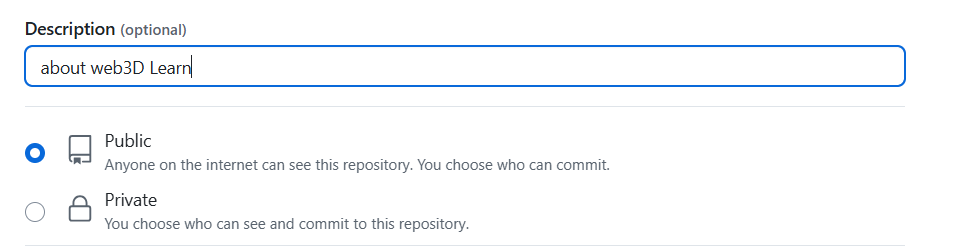
远程项目我们要先在github上面创建一个仓库

在git主页点击这个就可以创建仓库了。

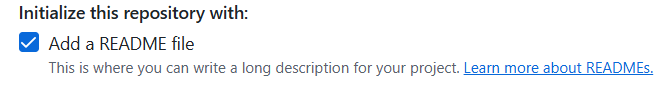
创建仓库这个也很简单。



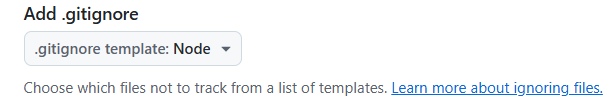
仓库名。



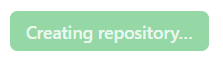
仓库的描述以及仓库是否公开。

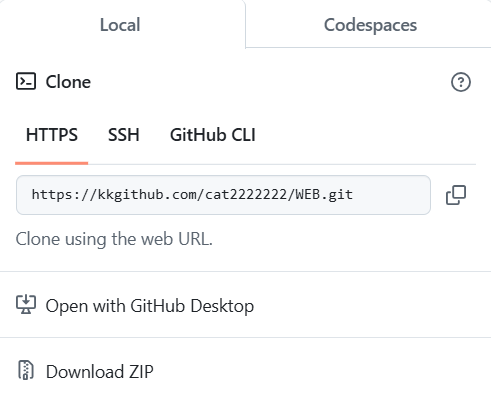


ReadMe文件。

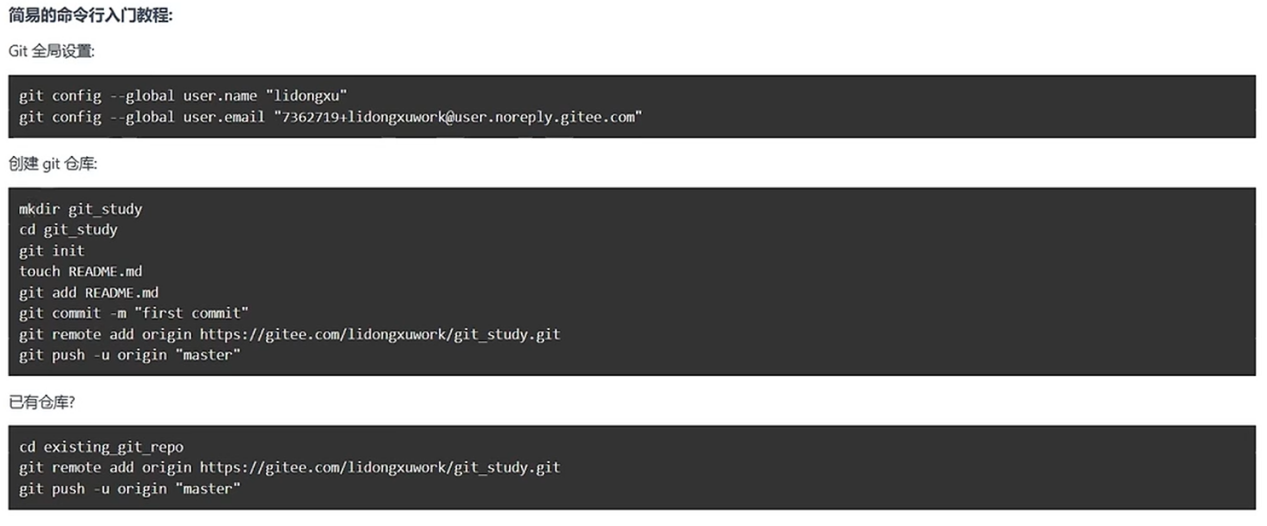


忽略文件。这些配置好之后就可以创建仓库了。



之后他会给你http和SSH两种方式,但是HTTPS这个要生成一个token不能一劳永逸所以我们用SSH来进行。原理都是一样的。

电脑上没有SSH要在里添加之后再到github的setting里面配置SSH才能用。



通过以上代码就能传到远程仓库了。

#### 克隆

本地没有文件我们要用**git clone 仓库地址**把文件克隆到本地。

#### 拉取

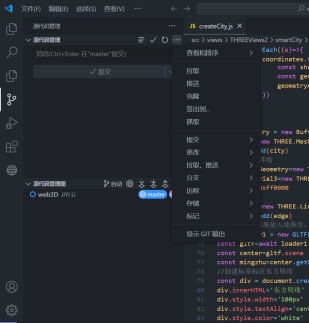
本地已有仓库但是远端仓库更新了,这种情况用**git pull 仓库别名 分支:分支**这种方法来拉取代码。

**大文件**



### VScode中使用

在VScode里其实有源代码管理工具,其实这种就是图形化界面调试,GUI。不用一直输命令行,会方便很多。



### Git常用命令





