

本科学生设计性实验报告

编者：吁丹

项目组长 詹建辉 学号 0174137

成员 吁丹 0174045 权佳欣 0174123

毛帅 0174135 黄友民 0174142

专业 软件工程

实验项目名称 Git使用操作说明

课程名称 软件工程综合实训

目录

[1 git的下载与安装（权佳欣） 1](#_Toc33305018)

[1.1 git的下载 1](#_Toc33305019)

[1.2 git的安装 2](#_Toc33305020)

[2 git的使用 10](#_Toc33305021)

[2.1 git用户信息配置（权佳欣） 10](#_Toc33305022)

[2.2 git工作基本流程（权佳欣） 10](#_Toc33305023)

[2.2.1 git工作区域 10](#_Toc33305024)

[2.2.2 向git仓库中添加文件 11](#_Toc33305025)

[2.3 git创建版本库（黄友民） 11](#_Toc33305026)

[2.4 git的基本操作（黄友民） 13](#_Toc33305027)

[2.4.1 将文件添加到暂存区 14](#_Toc33305028)

[2.4.2 将文件添加到仓库 14](#_Toc33305029)

[2.4.3 查看工作目录和暂存区的状态 15](#_Toc33305030)

[2.4.4 查看文件修改的内容 15](#_Toc33305031)

[2.4.5 查看历史记录和版本回退 16](#_Toc33305032)

[2.4.6 撤销修改和删除文件操作 17](#_Toc33305033)

[2.5 git的分支（毛帅） 19](#_Toc33305034)

[2.5.1 创建与合并分支 19](#_Toc33305035)

[2.5.2 删除分支 21](#_Toc33305036)

[2.5.3 解决冲突 21](#_Toc33305037)

[2.5.4 分支管理策略 22](#_Toc33305038)

[2.5.5 工作现场的隐藏及还原 24](#_Toc33305039)

[2.6 git的远程仓库（詹健辉） 24](#_Toc33305040)

[2.7 多人协作（詹健辉） 31](#_Toc33305041)

[3 组员操作git记录 36](#_Toc33305042)

# 1 git的下载与安装（权佳欣）

## 1.1 git的下载

1. 进入git官网，在downloads处选择相应的版本下载，其官网地址为<https://git-scm.com/downloads>，如图1.1、图1.2所示。

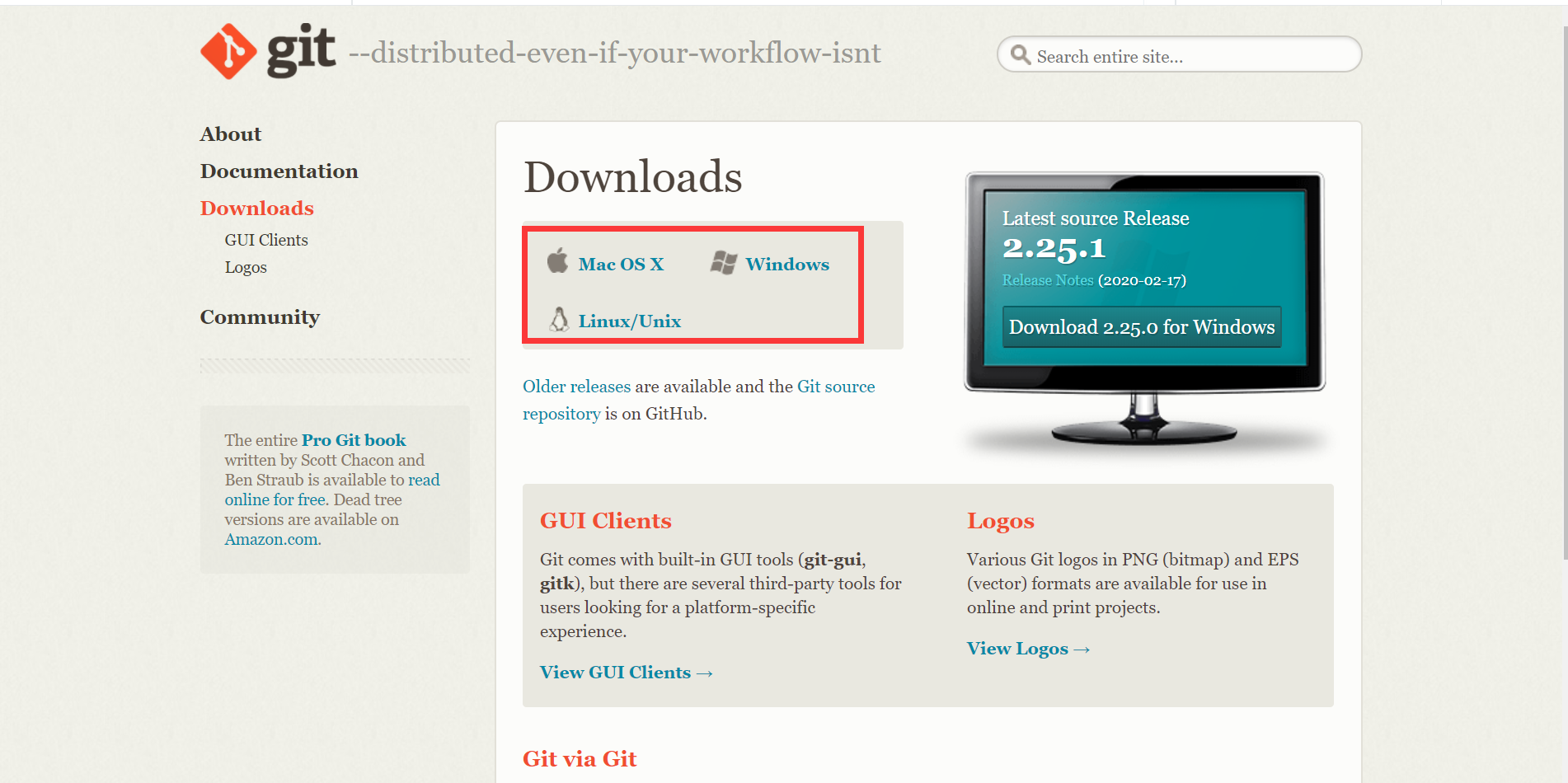


图1-1 git官网

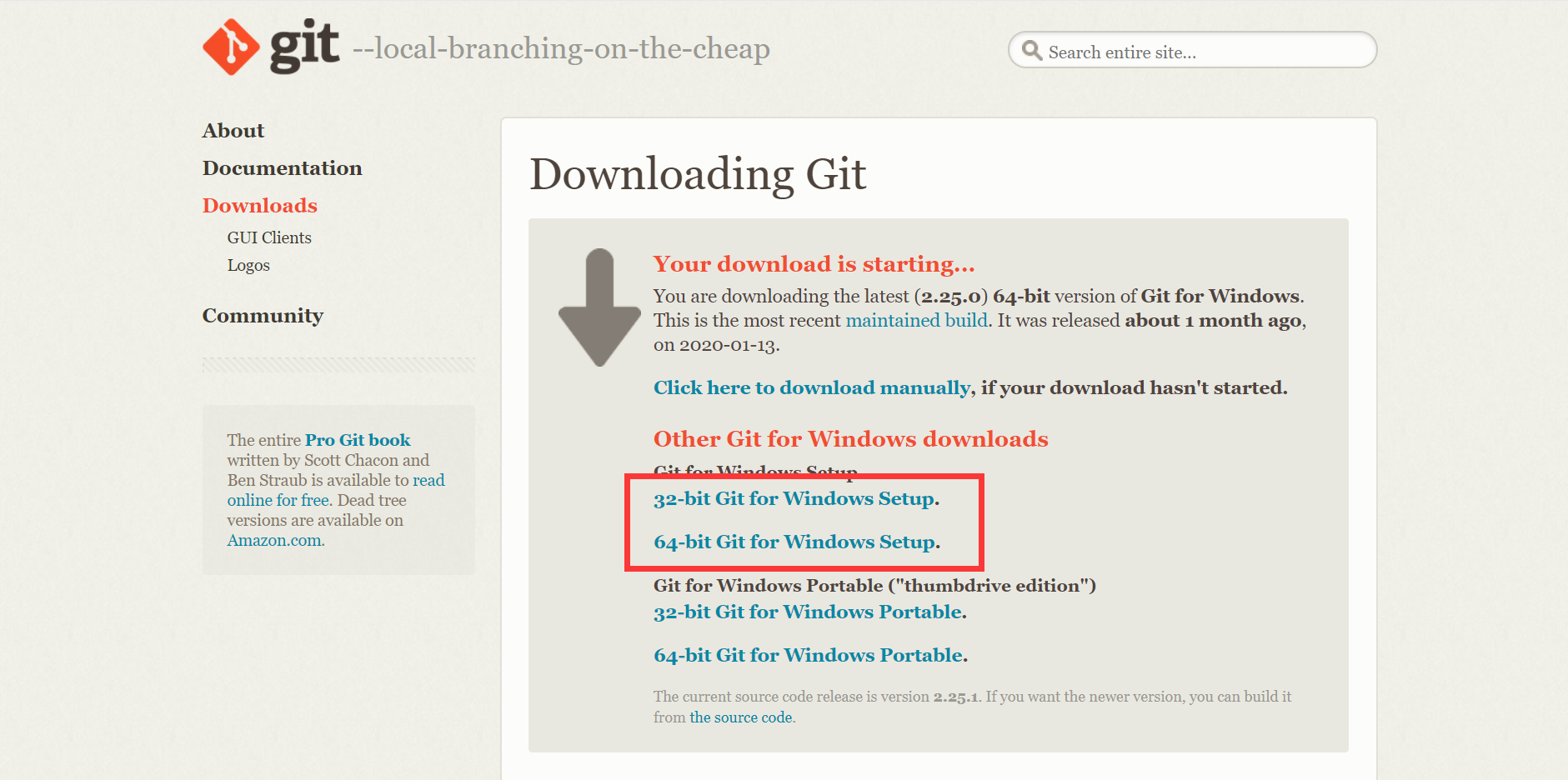


图1-2 git版本

1. 下载成功，如图1-3所示。



图1-3 git可执行文件

## 1.2 git的安装

1. 双击打开下载好的git可执行文件，打开安装设置界面，直接点击Next，其初始界面如图1-4所示。

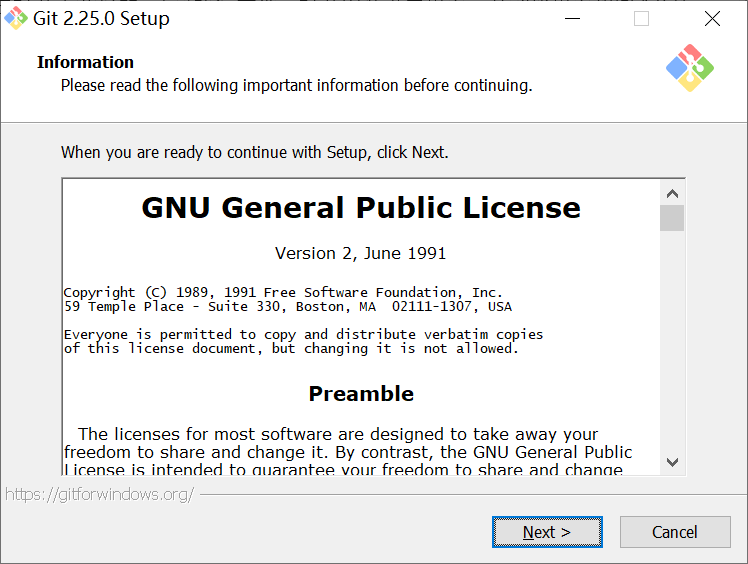


图1-4 git安装设置初始界面

1. 选择安装目录（一般情况下不使用默认盘），如图1-5所示。点击右方Browse按钮浏览选择即可。选择好目录后点击Next。

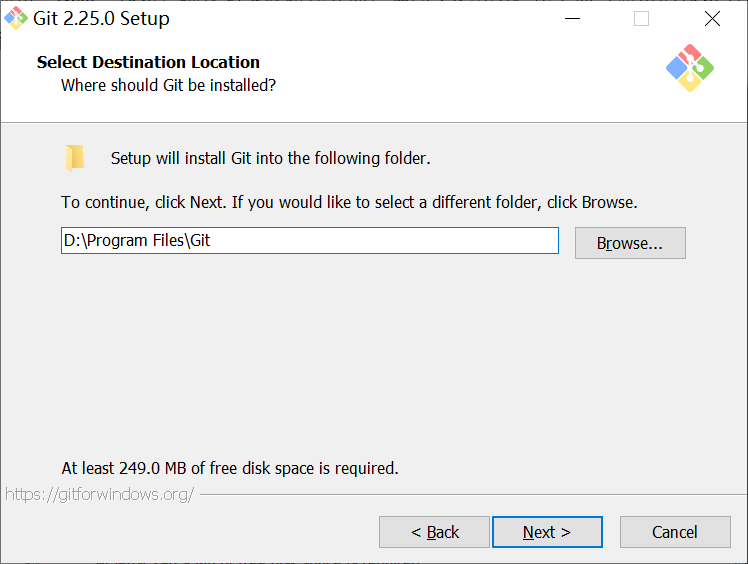


图1-5 安装目录选择界面

1. 选择组件安装，如图1-6所示。使用默认配置，点击Next。

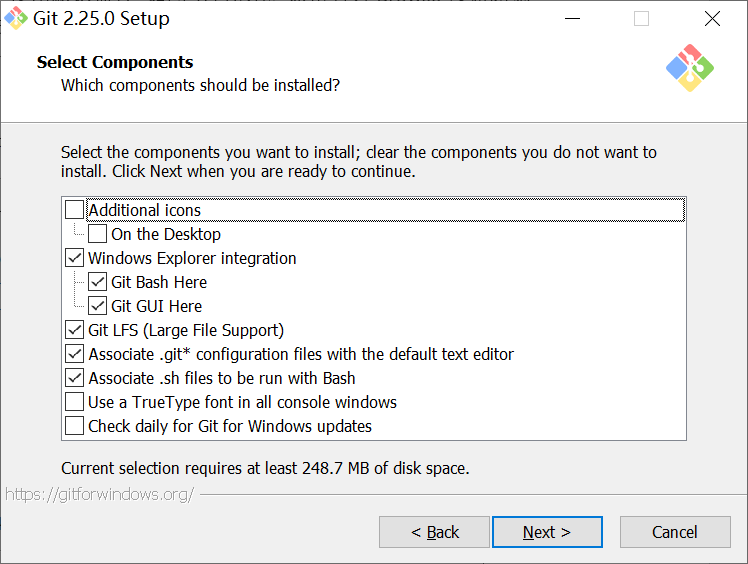


图1-6 安装组件选择界面

**说明：**

1. Additional icons：附加图标
2. On the Desktop：在桌面上
3. Windows Explorer integration Windows：资源管理器集成鼠标右键菜单
4. Git Bash Here：命令行模式
5. ​Git GUI Here：图形界面
6. Git LFS (Large File Support)：大文件支持
7. Associate .git\* configuration files with the default text editor：将 .git 配置文件与默认文本编辑器相关联
8. Associate .sh files to be run with Bash：将.sh文件关联到Bash运行
9. Use a TrueType font in all console windows：在所有控制台窗口中使用TrueType字体
10. Check daily for Git for Windows updates：每天检查Git是否有Windows更新
11. 选择是否在“开始”菜单栏创建快捷方式以及快捷方式的名称，如图1-7所示。使用默认配置，点击Next。

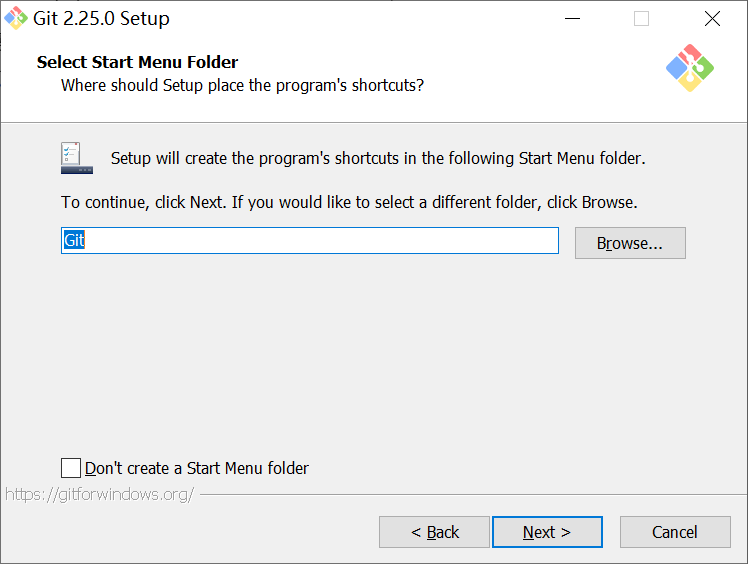


图1-7 创建开始菜单目录

1. 选择git的默认编辑器，如图1-8所示。使用默认配置，点击Next。

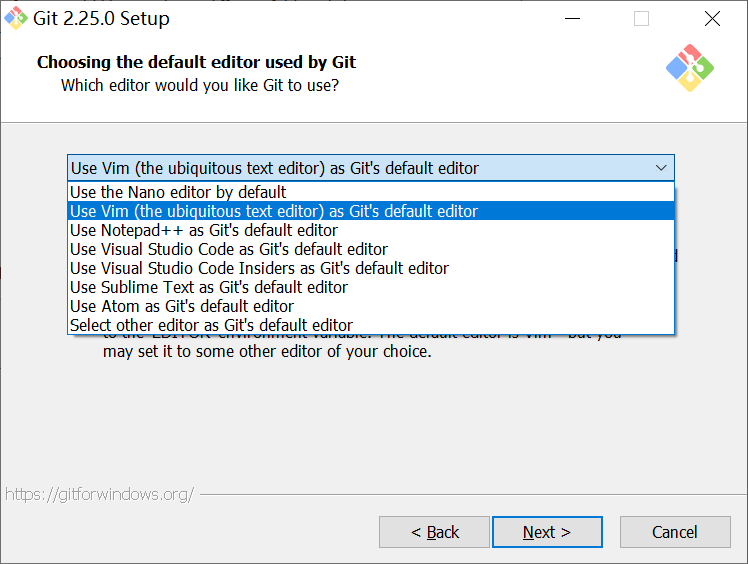


图1-8 git默认编辑器选择页面

1. 选择命令行工具，如图1-9所示。使用默认配置，点击Next。

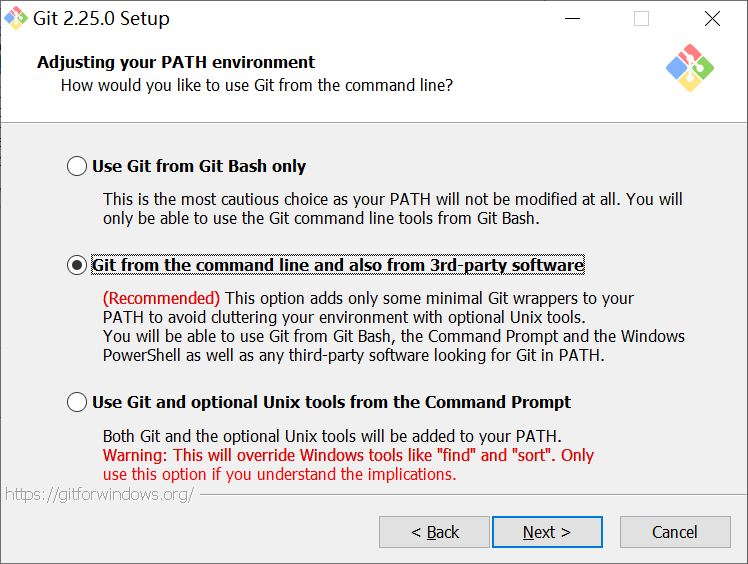


图1-9 命令行工具选择界面

1. 选择HTTPS传输后端，如图1-10所示。选择默认配置，点击Next。

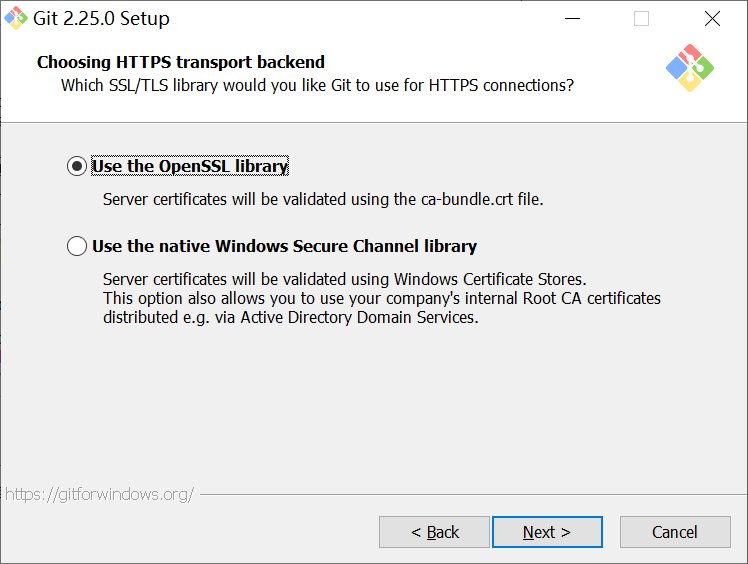


图1-10 HTTPS传输后端选择

1. 配置行结束转换，如图1-11所示。选择默认配置，点击Next。

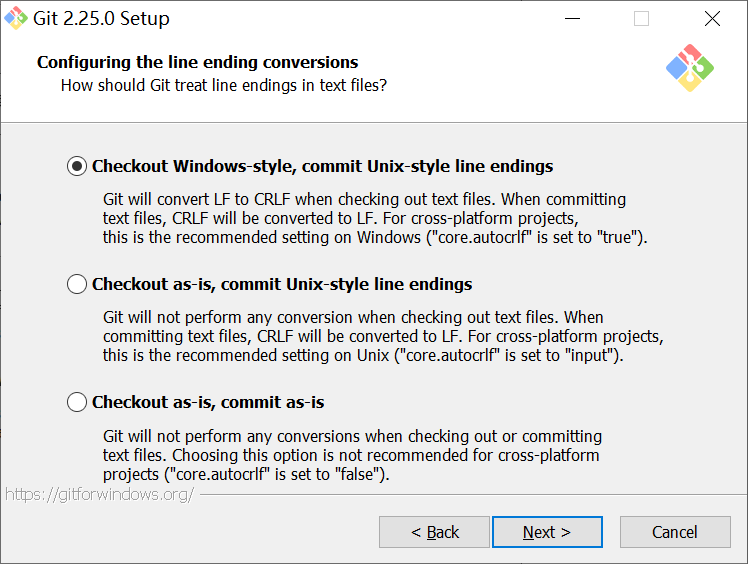


图1-11 配置行结束转换

**说明：**

1. 在检出文本文件时，Git会将LF转换为CRLF。当提交文本文件时，CRLF将转换为LF。 对于跨平台项目，这是Windows上推荐的设置（“core.autocrlf”设置为“true”）
2. 在检出文本文件时，Git不会执行任何转换。 提交文本文件时，CRLF将转换为LF。 对于跨平台项目，这是Unix上的推荐设置 （“core.autocrlf”设置为“input”）
3. 在检出或提交文本文件时，Git不会执行任何转换。对于跨平台项目，不推荐使用此选项（“core.autocrlf”设置为“false”）
4. 配置终端模拟器以与Git Bash一起使用，如图1-12所示。选择默认配置，点击Next。

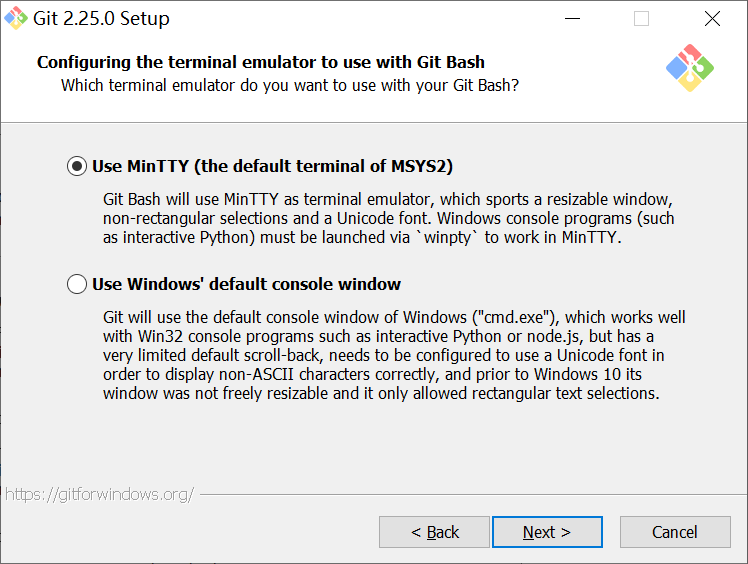


图1-12 配置终端模拟器页面

1. 配置额外的选项，如图1-13所示。选择默认配置，点击Istall安装。

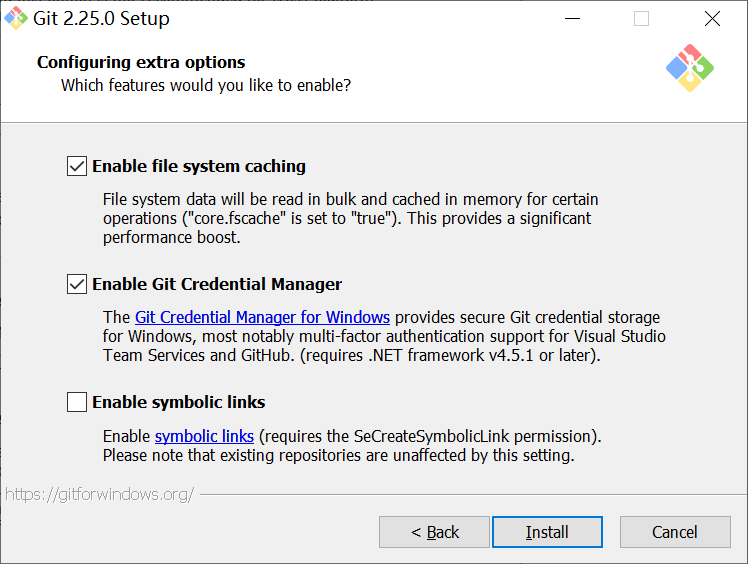


图1-13 配置额外的选项页面

1. 等待安装完成。如图1-14所示。

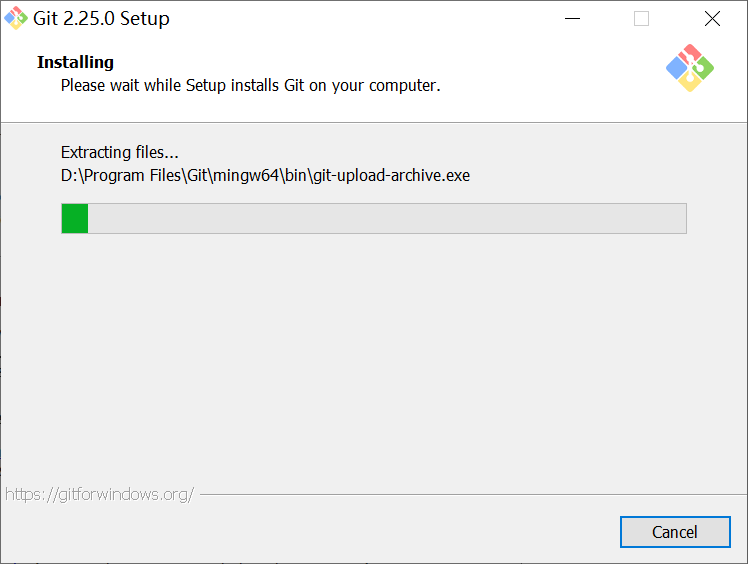


图1-14 正在安装

1. 安装完成。如图1-15所示。

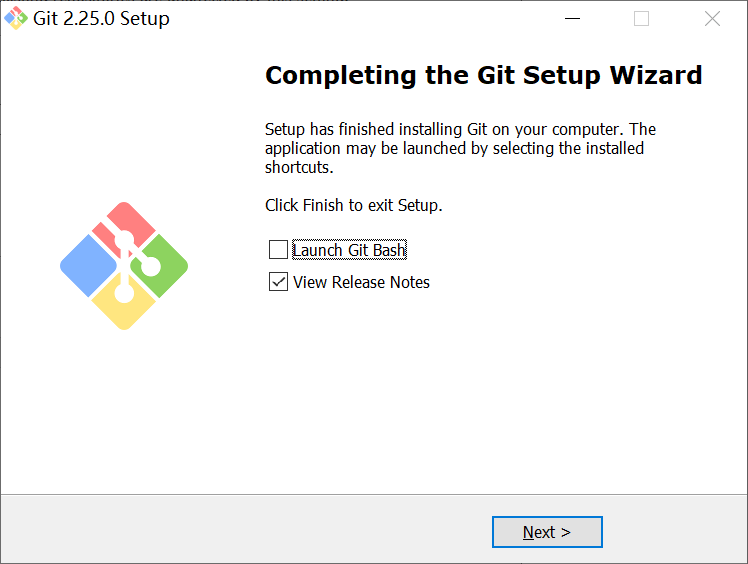


图1-15 安装完成

1. 验证是否安装成功。

方法一：回到桌面，点击右键。若有Git GUI Here和Git GUI Bash选项，则安装成功。

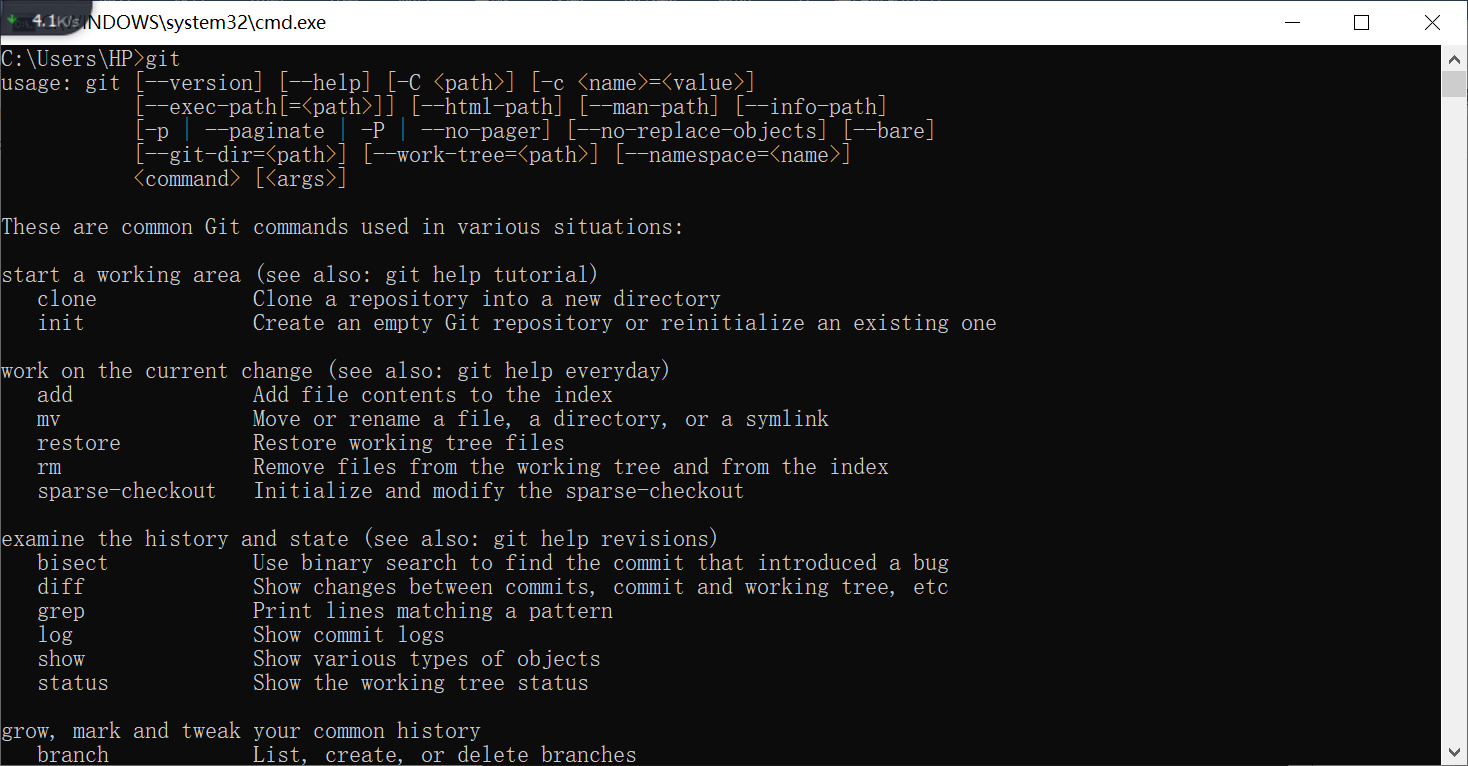
方法二：进入cmd命令界面，输入git，若出现如下图1-16所示页面，则安装成功。

图1-16 检验git是否安装成功

# 2 git的使用

## 2.1 git用户信息配置（权佳欣）

用户名和邮箱是git本地客户端的一个变量，不随git库的改变而改变，每次commit都会用到用户名和邮箱记录。

1. 设置用户名和邮箱

双击打开Git Bash，输入以下指令来配置用户名和邮箱，如图2-1所示：

git config --global user.name “与GitHub上一致的用户名”

git config --global user.email “与Github上一致的邮箱”

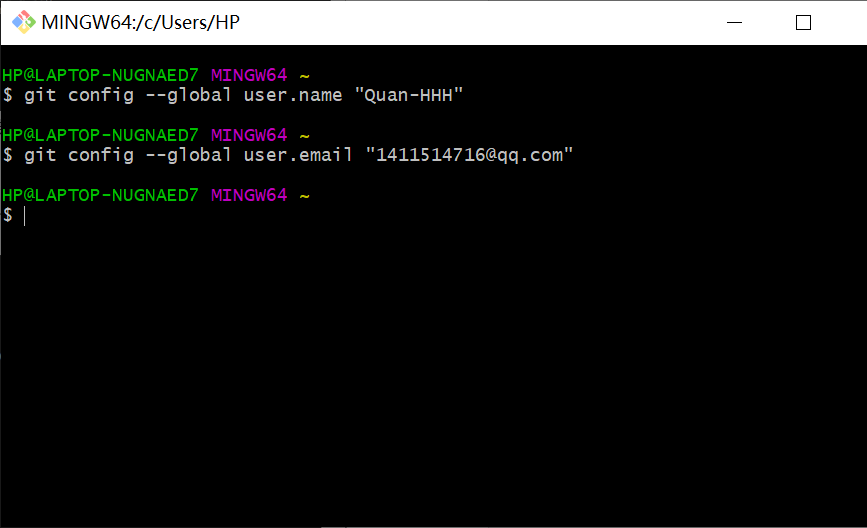


图2-1配置用户名和邮箱

1. 修改用户名和邮箱

双击打开Git Bash，输入以下指令来修改用户名和邮箱：

git config user.name

git config user.email

1. 查看用户名和邮箱

双击打开Git Bash，输入以下指令来查看用户名和邮箱：

git config --global user.name "username"

git config –global user.email “email”

## 2.2 git工作基本流程（权佳欣）

### 2.2.1 git工作区域

Git有包括Git Repository(Git仓库)、暂存区和工作区(Working Dictionary)在内的三个工作区域。其中Git仓库用来将最终确定的文件保存到仓库，成为一个新的版本，并对他人可见；暂存区用来暂存已经修改的文件，最后统一提交到git仓库中；工作区实现添加、编辑、修改文件等操作。

### 2.2.2 向git仓库中添加文件

向git仓库添加文件流程如下图2-2所示。

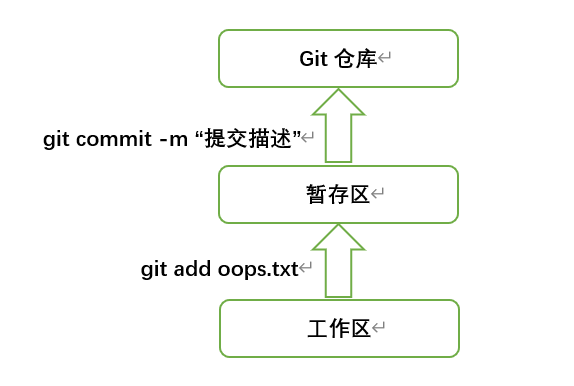


图2-2 向git仓库添加文件

## 2.3 git创建版本库（黄友民）

命令：git init

通过git init 语句可以再某个目录种生成一个版本库，目录下会多了一个.git的文件夹，这个目录是git来跟踪管理版本的，不要随便动这个目录里面的文件，否则会把git仓库给破坏了。选择路径有两种方法：

**方法一：在桌面打开 Git Bash，通过”cd + 路径”和“mkdir + 文件夹”来到你输入的路径和创建文件夹，如图2-3所示。**

**方法二：是直接打开文件夹，鼠标邮件，点击Git Bash Here就可以在当前目录打开git，如图2-4和2-5所示。**



图2-3 通过方法一建立版本库



图2-4 打开当前目录的git bash

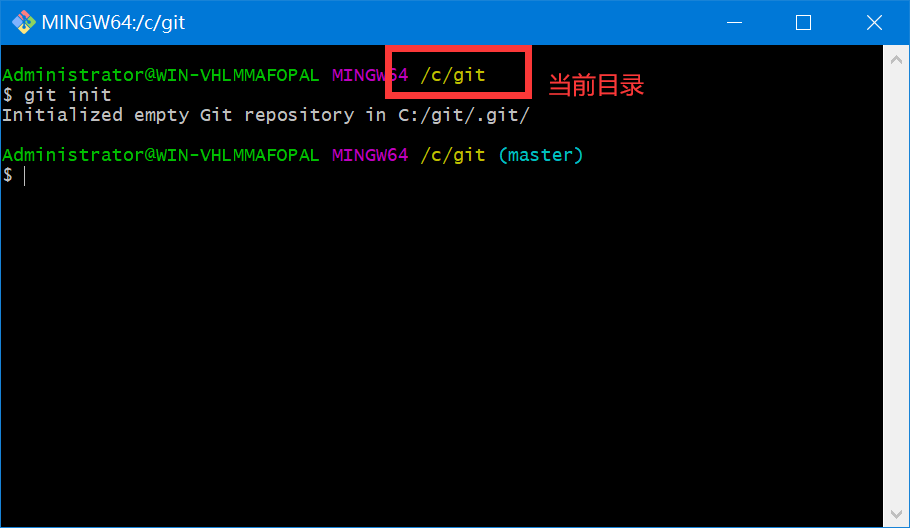


图2-5 方法二创建版本库

需要注意的是，.git 文件夹是隐藏的，如果要看到，需要在文件夹选项中设置。在 Windows 7 操作系统中，通过“计算机-->组织-->文件夹和搜索选项-->查看-->显示隐藏的文件、文件夹和驱动器”可以显示隐藏文件夹。在 Windows 10 操作系统中，通过“此电脑-->查看-->在复选框中勾选隐藏的项目”即可显示隐藏文件夹。

## 2.4 git的基本操作（黄友民）

为了更好地对命令的功能进行演示，在刚刚新建的版本库的路径（即.git 文件夹所在的目录）下创建一个文本文档 test.txt，文本文档的内容为一行“111111”，如图2-6所示。

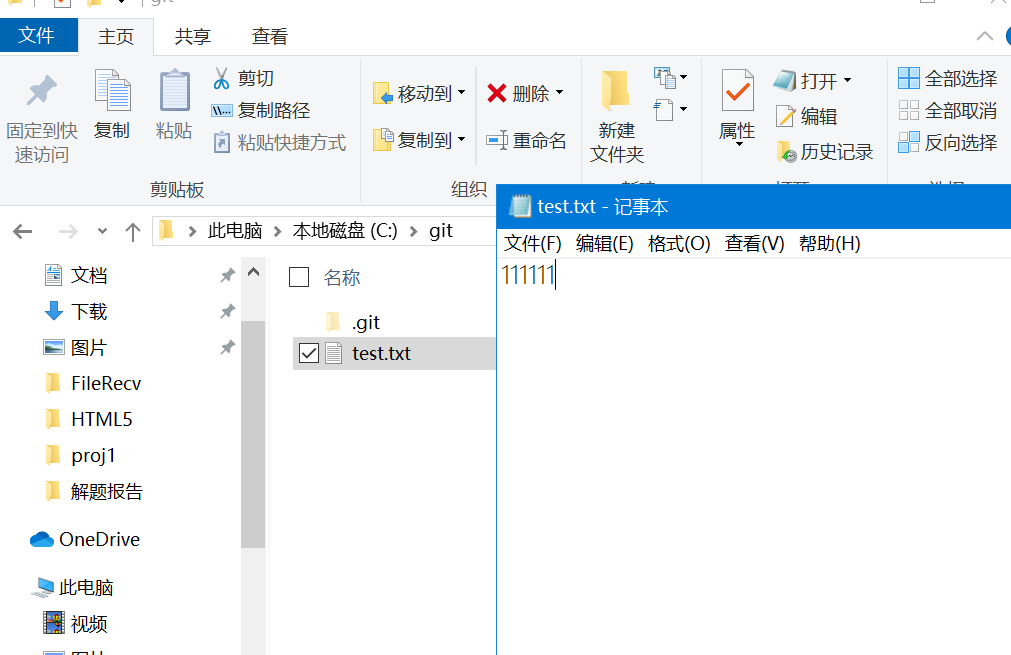


图2-6创建文本文档

### 2.4.1 将文件添加到暂存区

命令：git add 文件名

通过这个命令，可以将文件添加到暂存区，如果添加之后没有报错，则说明添加到暂存区成功，如图2-7所示。

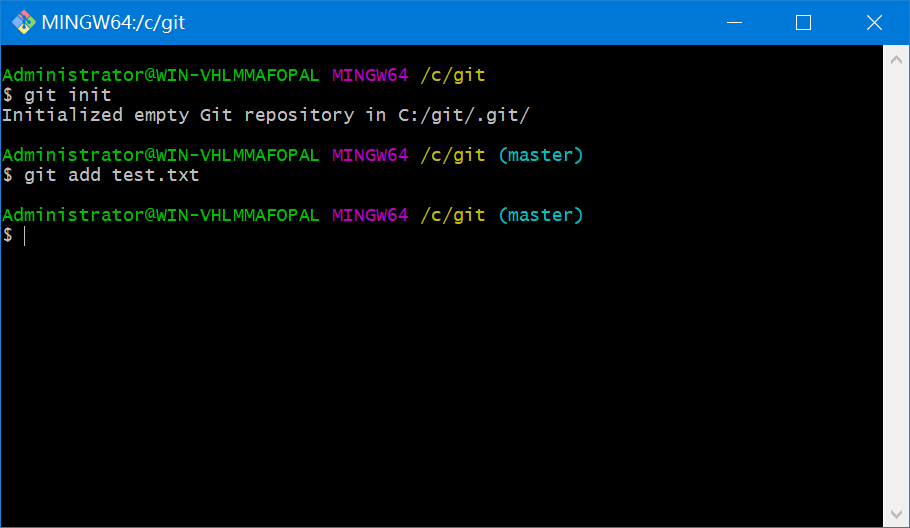


图2-7 将文件添加到暂存区

### 2.4.2 将文件添加到仓库

命令：git commit -m “备注信息（即为这次提交这个动作取个名字）”

通过这个命令，可以将暂存区中的文件添加到仓库中。其中引号中的备注信息，表示对这次提交的备注信息，让你下次查看的时候，能够直观地知道这次提交的是什么文件，如图2-8所示。

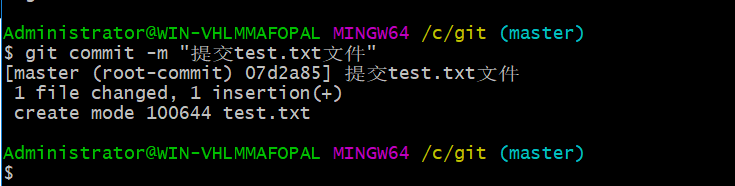


图2-8 将文件添加到仓库

### 2.4.3 查看工作目录和暂存区的状态

命令：git status

通过这个命令，可以查看工作目录和暂存区的状态。现在使用该命令查看，会发现没有可改变的状态，如图2-9所示。

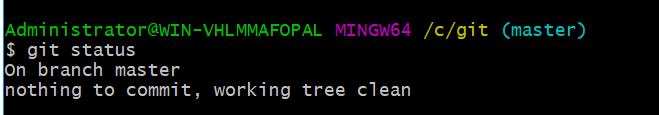


图2-9 查看工作目录和暂存区的状态

然后在test.txt文件中加一行“222222“，再在git bash中通过该命令查看状态，会发现提示了工作目录有改动，如图2-10所示，通过该命令告诉我们test.txt文件已被修改，但是是未被提交的修改。



图2-10 查看文件修改状态

### 2.4.4 查看文件修改的内容

命令：git diff 文件名

使用该命令可以查看未提交的修改后的文件和存在于仓库中的同一文件的差异，即查看文件修改的内容，如图2-11所示。

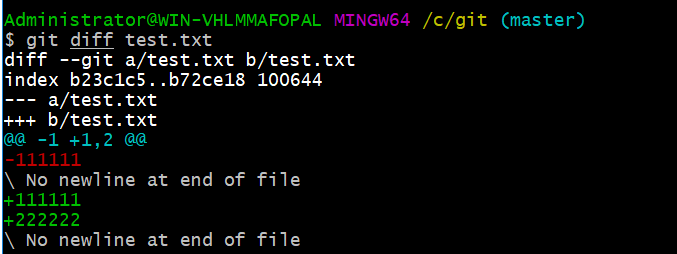


图2-11 查看文件修改的内容

了解到修改前后的内容之后，可以将修改后的文件也发送到仓库中，步骤见 2.4.1 和2.4.2。为了便于操作，再在test.txt中第三行添加一行字母 aaaaaaaa，并提交到仓库，如图2-12所示。

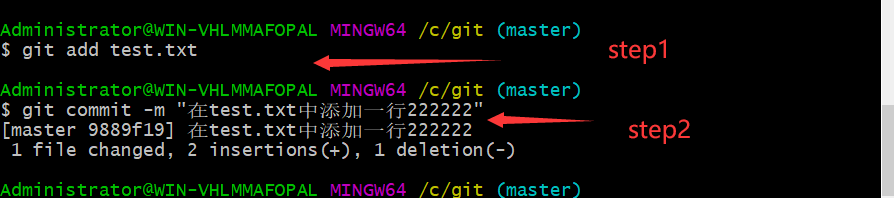


图2-12 将修改后的文件提交到仓库

### 2.4.5 查看历史记录和版本回退

**查看历史记录**：

命令：git log 或者 git log --pretty=oneline

其中使用 git log 可以查看历史记录的版本信息，信息显示比较全同时比较乱，而 git log --pretty=oneline 则是每个历史记录在显示的信息更简洁。我们可以看到最近三次提交，如图2-13 所示。

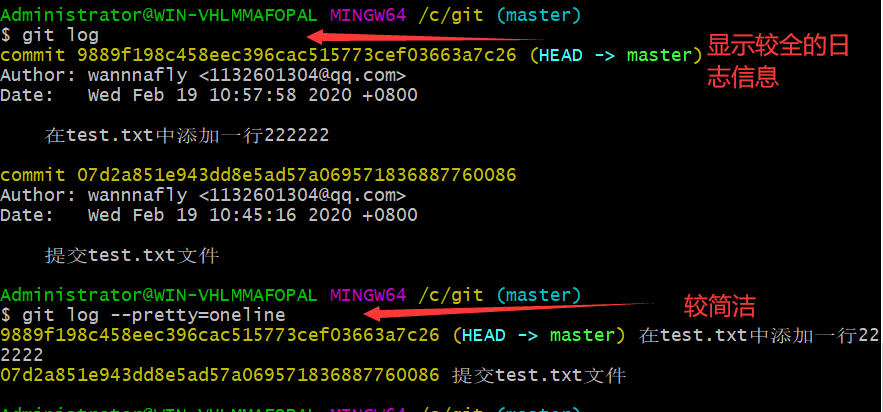


图2-13 查看历史记录

**版本回退：**

命令：git reset –hard HEAD^ 或者git reset –hard HEAD^^或者git reset –hard HEAD~数字

通过这些命令，可以将版本回退到上个版本、上上个版本或者指定的某个版本。如图2-14 所示。如果报错，把index.lock这个文件删掉。

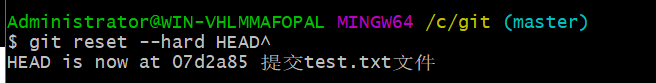


图2-14 查看历史记录

通过cat 文件名可以查看当前版本的 test.txt 中的内容。如图2-15所示。

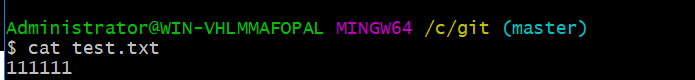


图2-15 版本回退

通过git log可调查日志，发现刚刚增加的“在test.txt中添加一行222222“不见了，如图2-16所示。



图2-16 通过git log调查日志

可以通过回退到某个版本号来恢复到增加第三行aaaaaaaa的内容。通过git reflog来获得版本号，并通过“git reset --hard 版本号”来回退到有 aaaaaaaa 的版本，如图 2-17 所示。

需要注意的是，这里涉及的所有版本回退即修改，都是针对工作目录的，即版本更改之后，可以看到在文件夹中的 test.txt 文件中的内容也会相应改变。

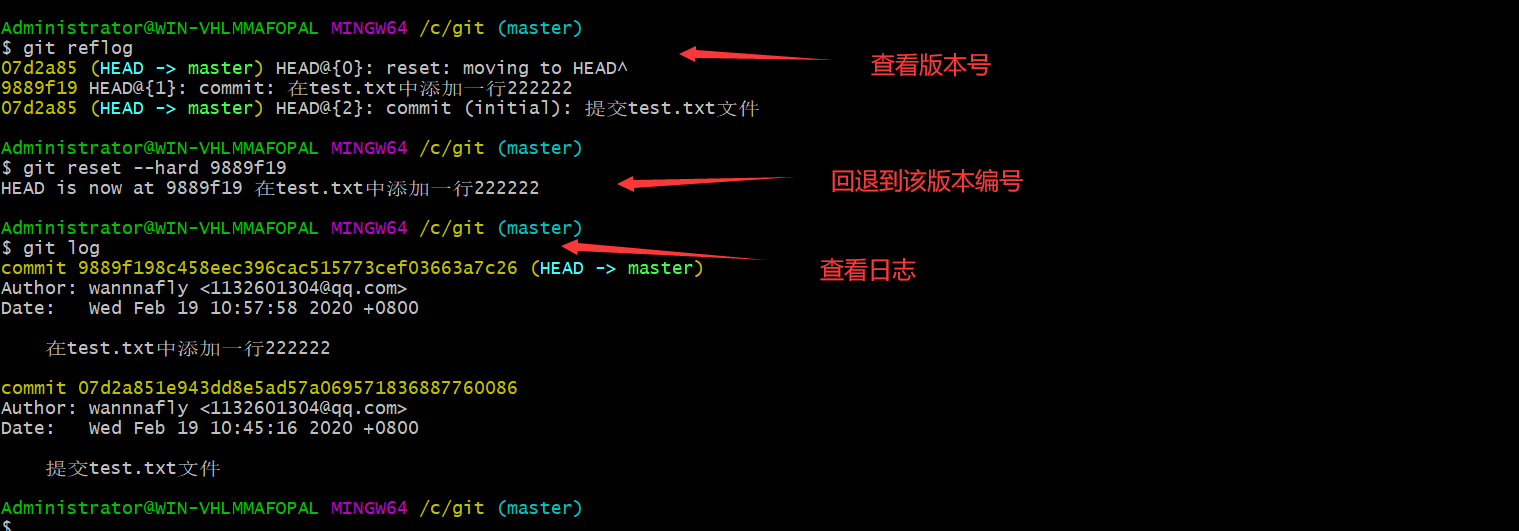


图2-17 通过版本号获取对应版本内容

### 2.4.6 撤销修改和删除文件操作

**撤销修改：**

命令：git checkout -- test.txt

把 test.txt 文件在工作区做的修改全部撤销，这里有两种情况如下：

情况一：test.txt 自动修改后，还没有放到暂存区，使用撤销修改就回到和版本库一模一样的状态。比如，在 test.txt 文件中，再加一行 333333并保存，然后在 Git Bash 中输入 git checkout -- test.txt，再打开 test.txt 可以发现刚刚加的一行333333没有了，说明撤销了刚刚的修改，如图 2-18 所示。

情况二：test.txt已经放入暂存区中，接着又做了修改，撤销修改就回到添加暂存区后的状态，如图2-19所示。

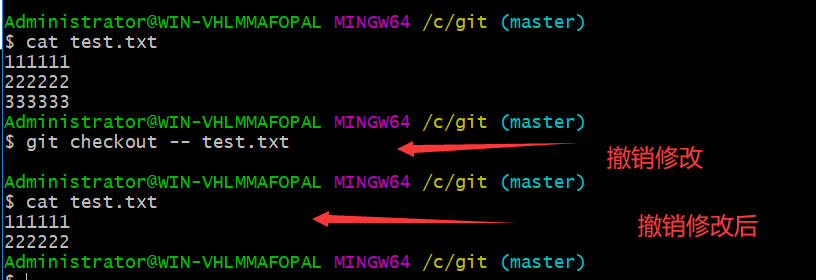


图2-18 撤销修改情况一



图2-19 撤销修改情况二

**删除文件：**

命令：rm 文件名（要包括扩展名）

通过这个命令可以删除文件，如图2-20所示。



图2-20 删除文件

## 2.5 git的分支（毛帅）

分支就是最后一个commit的引用，每一个commit都引用着前一个commit，这就构成了一个链条，也就是分支。每一个git仓库都默认维护着者一个分支master（主分支），master分支是所有开发活动的核心分支，所有的开发活动产生的输出物最终都会反映到主分支的代码中，master分支应该始终和生产环境保持一致。

### 2.5.1 创建与合并分支

创建分支：git branch 分支名称

切换到该分支：git checkout 分支名称

创建分支并转到该分支：git checkout -b 分支名，如图2-21所示。

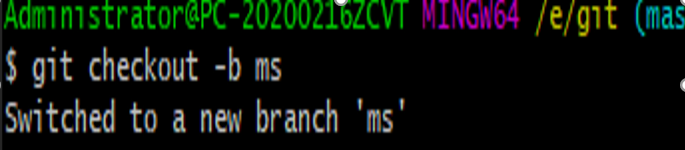


图2-21 创建分支并转到该分支

查看当前所有分支：git branch，且在当前分支名前有“\*”号标注，如图2-22所示。



图2-22 查看当前所有分支

合并分支：git merge 分支名（在该分支名的分支中使用）先在ms分支中使用cat test.txt来查看其中内容，如图2-23所示。

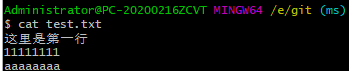


图2-23 合并分支

在test.txt文件中加一行33333333，如图2-24所示。

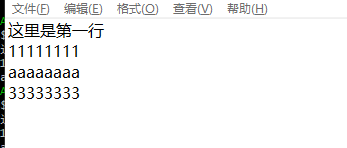


图2-24 向test.txt文件中添加内容

再次查看并提交，如图2-25所示。

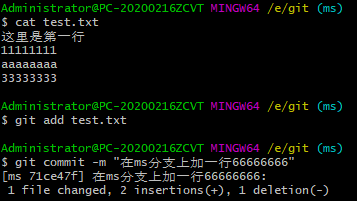


图2-25 查看并提交

切换到master分支并查看，可以看到并没有我们刚刚加进去的33333333，如图2-26所示。

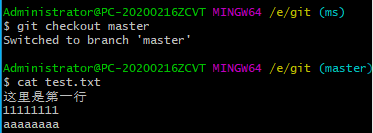


图2-26 切换到master分支查看

我们使用合并命令，将master和ms分支合并，然后再次查看，就可以看到我们新加入的33333333了，如图2-27所示。

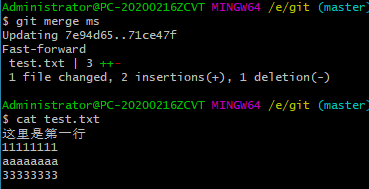


图2-27 master和ms分支合并

### 2.5.2 删除分支

删除分支可以使用git branch -d 分支名来删除，如图2-28所示。

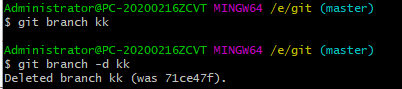


图2-28 删除分支

### 2.5.3 解决冲突

当我们在两个分支上都修改了文件，那么在合并的时候会产生冲突，如我们在master分支上加了个“2”并上传到仓库，在新建的g分支上加了个“1”并上传到仓库，那么在合并的过程中会产生冲突，如图2-29和图2-30所示。

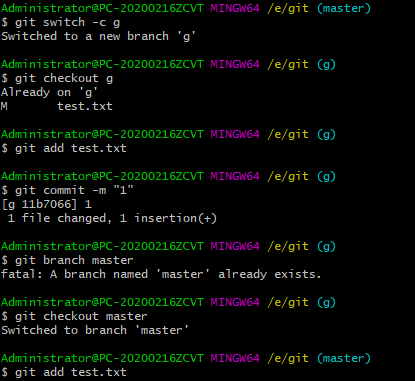


图2-29 在两个分支上修改文件

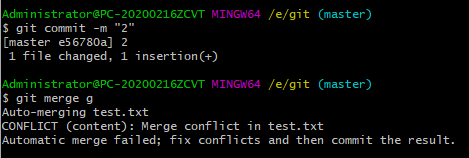


图2-30 合并分支时产生冲突

那么此时我们只需要修改g分支中test.txt的添加内容和master分支中的添加内容相同就行了，这时就没有冲突了，如图2-31所示。

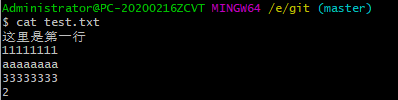


图2-31 正确合并分支结果

### 2.5.4 分支管理策略

一般合并分支时，git使用“Fast forward”模式，在这种模式下，删除分支后，会丢掉分支信息，现在我们来使用参数 - no-ff来禁用“Fast forward”模式，演示步骤如下：

步骤一：新建分支dev；

步骤二：修改test.txt的内容并上传；

步骤三：切换回主分支；

步骤四：合并dev分支，使用“git merge --no-ff -m”来注释dev；

步骤五：查看历史记录。

演示结果如图2-32和图2-33所示。

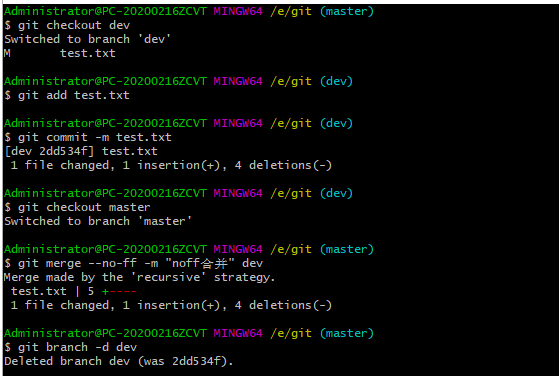


图2-32 步骤一~步骤四

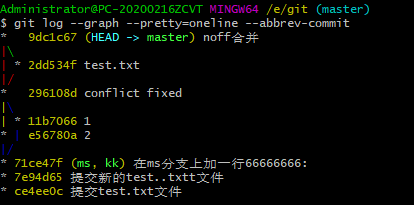


图2-33 步骤五及演示结果

其中2dd534f即为操作记录，所以说明使用参数“- no-ff”来禁用“Fast forward”是可行的。

### 2.5.5 工作现场的隐藏及还原

隐藏工作现场：git stash，如图2-34所示。

工作现场列表：git stash list，如图2-37所示。

恢复工作现场：git stash apply，如图2-37所示。

在我们的开发过程中，如果遇到了bug，可以先将我们的工作现场隐藏起来，通过建立临时分支来解决bug，然后再删除临时分支即可，如图2-35和图2-36所示



图2-34 隐藏工作现场



图2-35 查看状态为空



图2-36 创建临时分支q

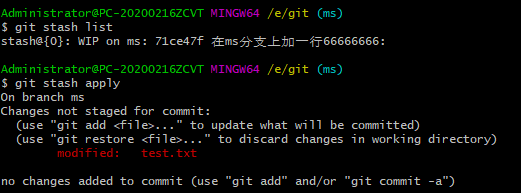


图2-37 查看状态列表并恢复工作现场

## 2.6 git的远程仓库（詹健辉）

**第一步：创建 SSH Key**。

在用户主目录（C：\Users\你的用户名）下，看看有没有.ssh 目录，一般第一次安装使用 Git 的系统是没有此文件夹的。如果没有则在 Git 安装目录中找到“ssh-keygen.exe”所在路径，并且添加到系统环境变

量 Path 中。然后在 Git Bash 中运行“ssh-keygen –t rsa –C “youremail@example.com”，后一路回车，按照提示输入文件的完整路径，如果直接回车就是默认在用户主目录中.ssh 文件夹中id\_rsa 和 id\_rsa.pub 的 SSH key 的密钥对，如图 2-38 所示。

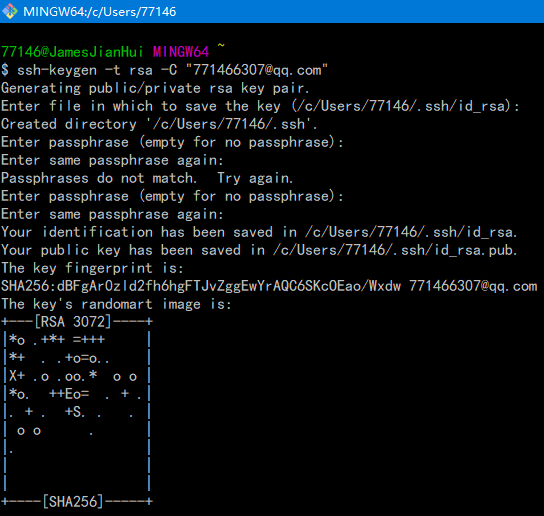


图 2-38 创建密钥

然后按照提示输入密钥的密码，显示创建的的结果和key fingerprint，如图 2-39所示。

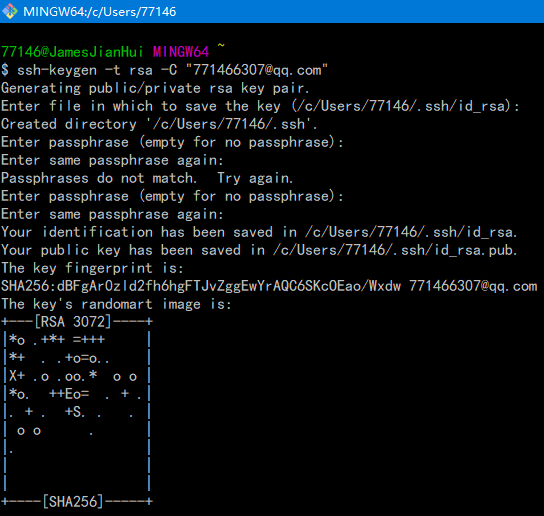


图 2-39 输入密钥的密码

id\_rsa 是私钥，不能泄露出去 id\_rsa.pub 是公钥。如果想要在.ssh 文件夹中生成其他名字的密钥，如“test”，可以执行如下命令：

ssh-keygen -t rsa -C "1805442532@qq.com" -f ~/.ssh/test，如图2-40所示。



图 2-40 在.ssh文件夹中创建其他名字的密钥

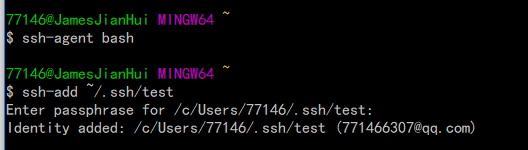
然后执行ssh-agent bash和ssh-add ~/.ssh/test，结果如图2-41所示。 

图2-41 执行结果

**第二步：登录 Github。**

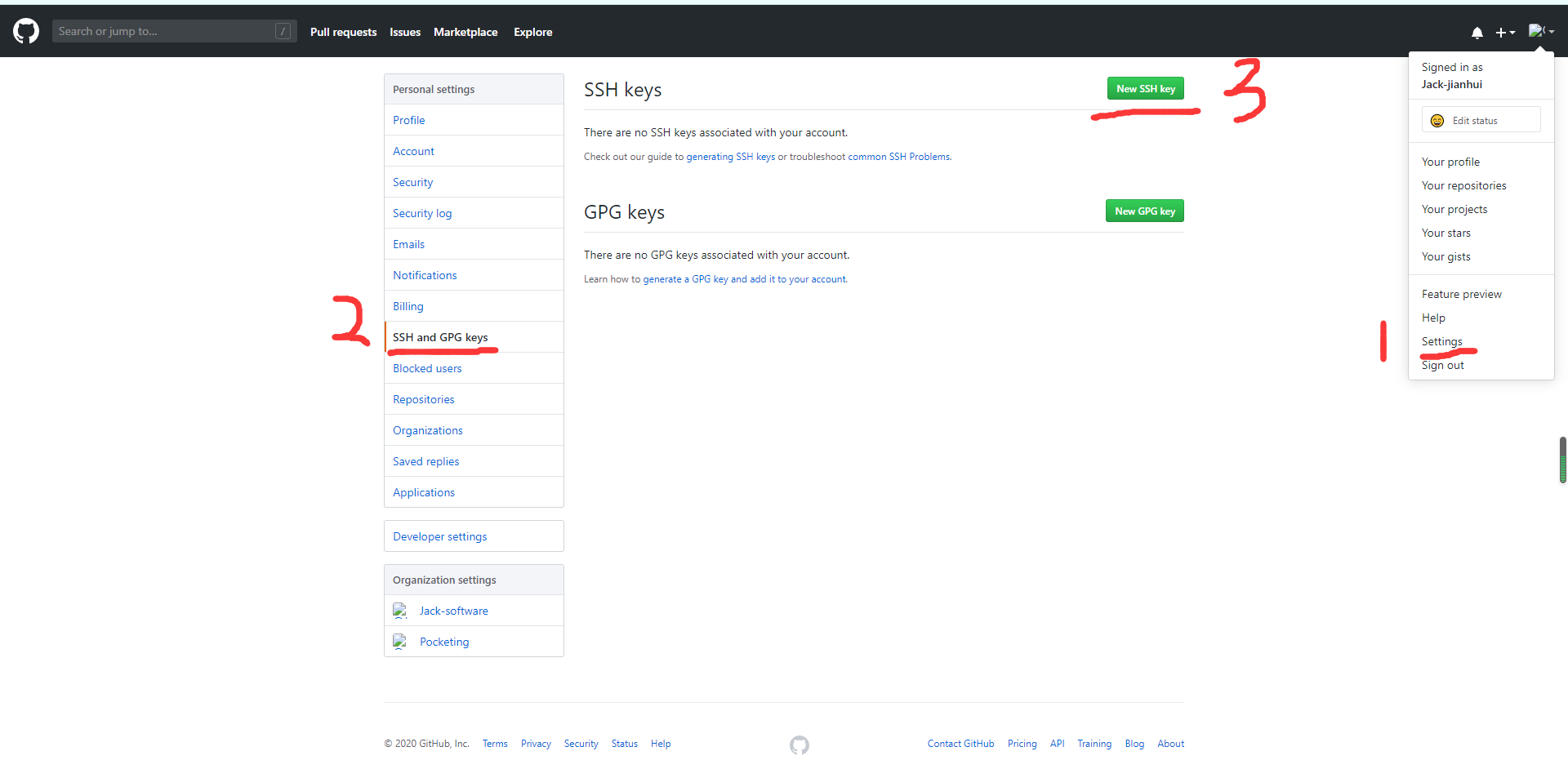
点击 Account settings，打开 SSH and GPG Keys 页面，新建SSH key，如图2-42所示。

图2-32 新建 SSH key

输入任意标题例如Test1，在 key 文本框中填入 id\_rsa.pub 文件中的内容，id\_rsa.pub 文件路径如图 2-43 所示，填入显示如图2-44所示。

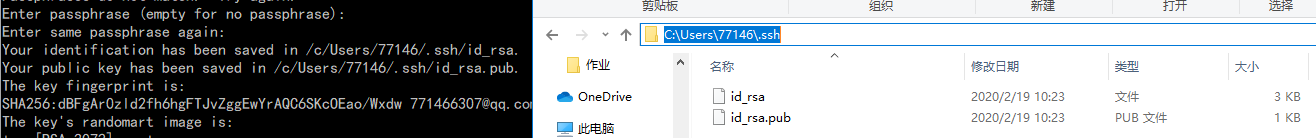


图2-43 id\_rsa.pub文件路径

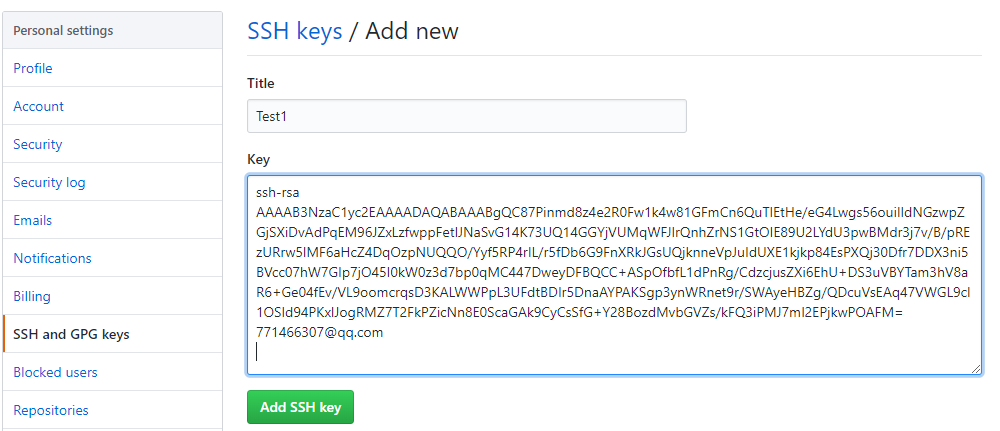


图2-44 设置标题和密钥

添加完成后如图2-45所示。

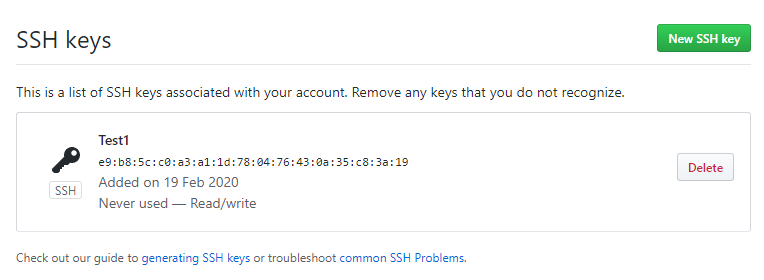


图2-45 新建SSH key完成

验证SSH是否和github连接上，可以输入命令：ssh -T [git@github.com](mailto:git@github.com)，其结果如图2-46所示。

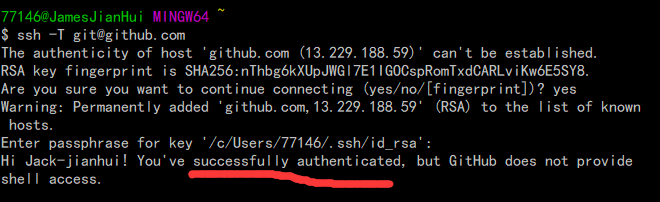


图2-46 测试连接github

**第三步：添加远程仓库并建立连接。**

填写好仓库名字和其他选项后，点击创建仓库。如图2-47和图2-48所示。

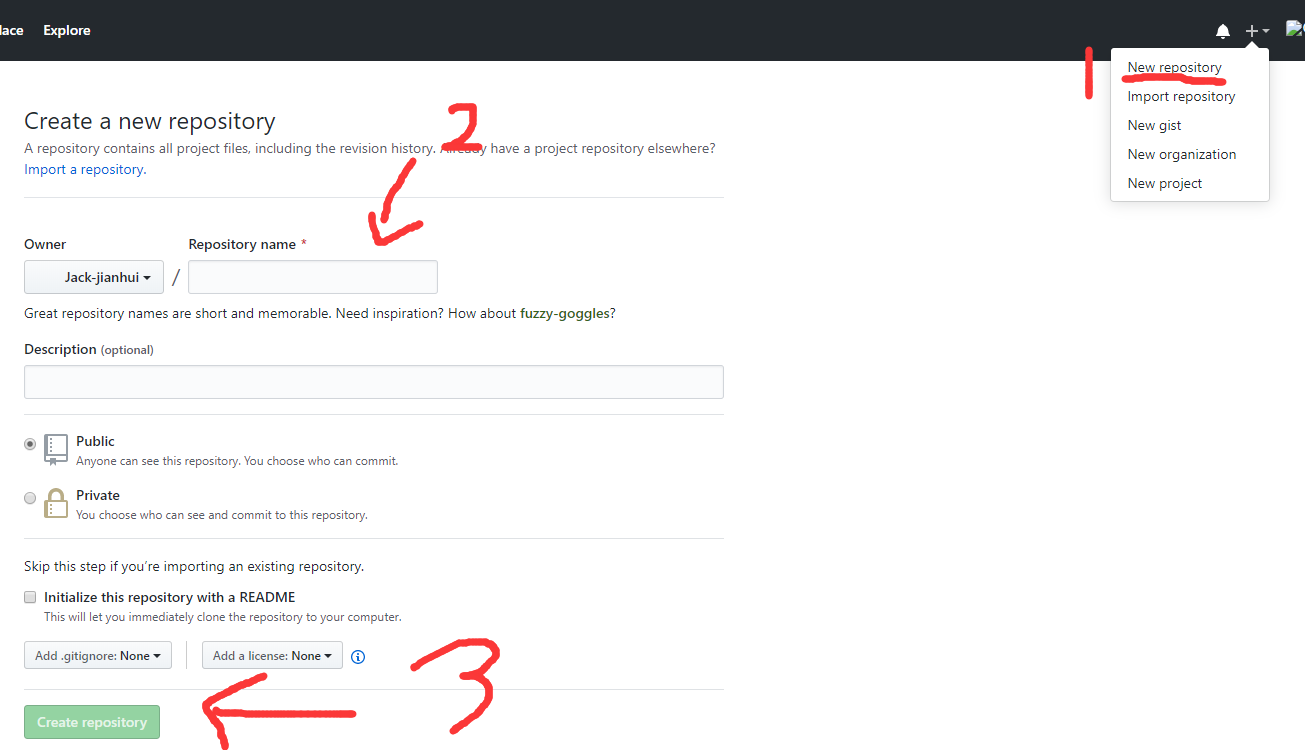


图2-47 新建仓库

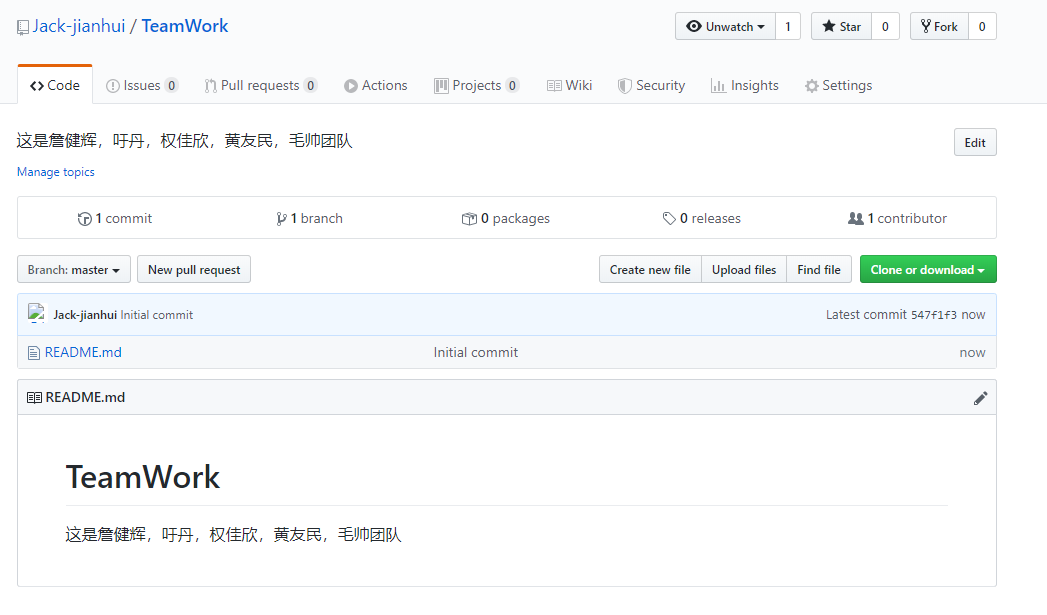


图2-48 仓库新建完成

然后新建提交内容，echo "第一次提交测试" >> demo.txt，将"第一次提交测试" 内容写入demo.txt文件中,如图2-49所示。

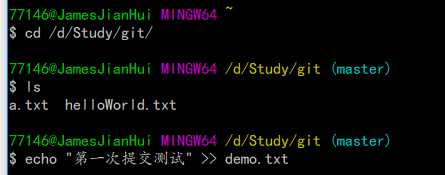


图2-49 新建提交内容

第一次提交时，首先进行初始化git版本库 git init，也就是当目录中没有.git文件夹时，需要git init。如果有，则不需要进行git init。添加文件到仓库中git add,但是文件暂存在缓存区，并没有提交到仓库中，如图2-50所示。

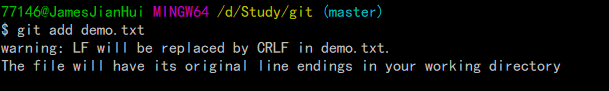


图2-50 添加文件到仓库

提交暂存的文件到gitHub上，使用git commit，如图2-51所示。

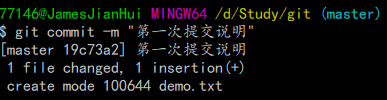


图2-51 提交文件

将本地更新的文件同步到远程仓库上，使用git remote add origin 远程仓库的地址，在github上选择ssh链接，如图2-52所示。

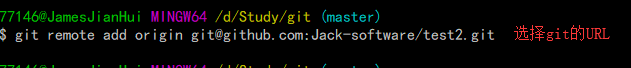


图2-52 更新到远程仓库上

使用git push -u origin master同步到github的远程仓库中。如果出现以下错误，则先执行git pull，然后执行git push --force origin master，如图2-53所示。  
注意：第一次git push -u origin master后，以后修改过后，就可以直接git push了。

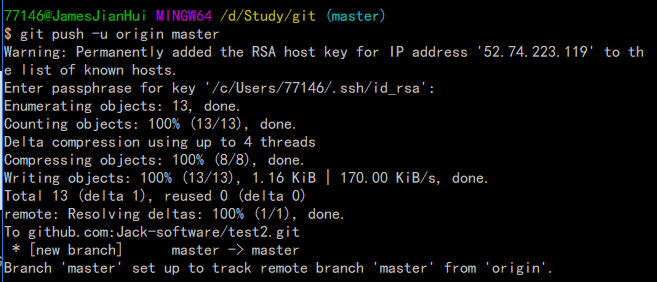


图2-53 同步到远程仓库中

git pull失败 ,提示：fatal: refusing to [merge](https://www.centos.bz/tag/merge/) unrelated histories

其实这个问题是因为 两个 根本不相干的 git 库， 一个是本地库， 一个是远端库， 然后本地要去推送到远端， 远端觉得这个本地库跟自己不相干， 所以告知无法合并。具体有两种方法：

**方法一：是从远端库拉下来代码，本地要加入的代码放到远端库下载到本地的库， 然后提交上去，因为这样的话，你基于的库就是远端的库，这是一次**[**update**](https://www.centos.bz/tag/update/)**了。**

**方法二:使用这个强制的方法git pull origin**[**master**](https://www.centos.bz/tag/master/)**--allow-unrelated-histories后面加上--allow-unrelated-histories，把两段不相干的分支进行强行合并后面再push就可以了[git push](https://www.centos.bz/tag/git-push/)** [**gitlab**](https://www.centos.bz/tag/gitlab/) **master:init。**

## 2.7 多人协作（詹健辉）

**第一步**，在 github 中我们运用它的 new organization（新建组织）功能来由项目组织者创建一个新的组织：传送点，如图 2-54 所示。

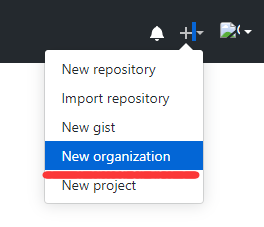


图2-54 选择创建新组织

接下来可以选择填写组织名以及 Email，选择项目所需的计划，我们这里选择第一个免费的（私人项目需要付费），然后进入到下一步，如图2-55 所示。

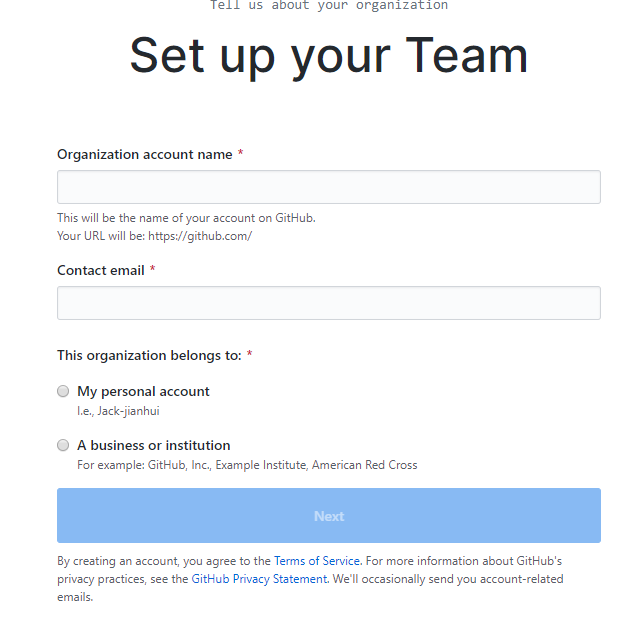
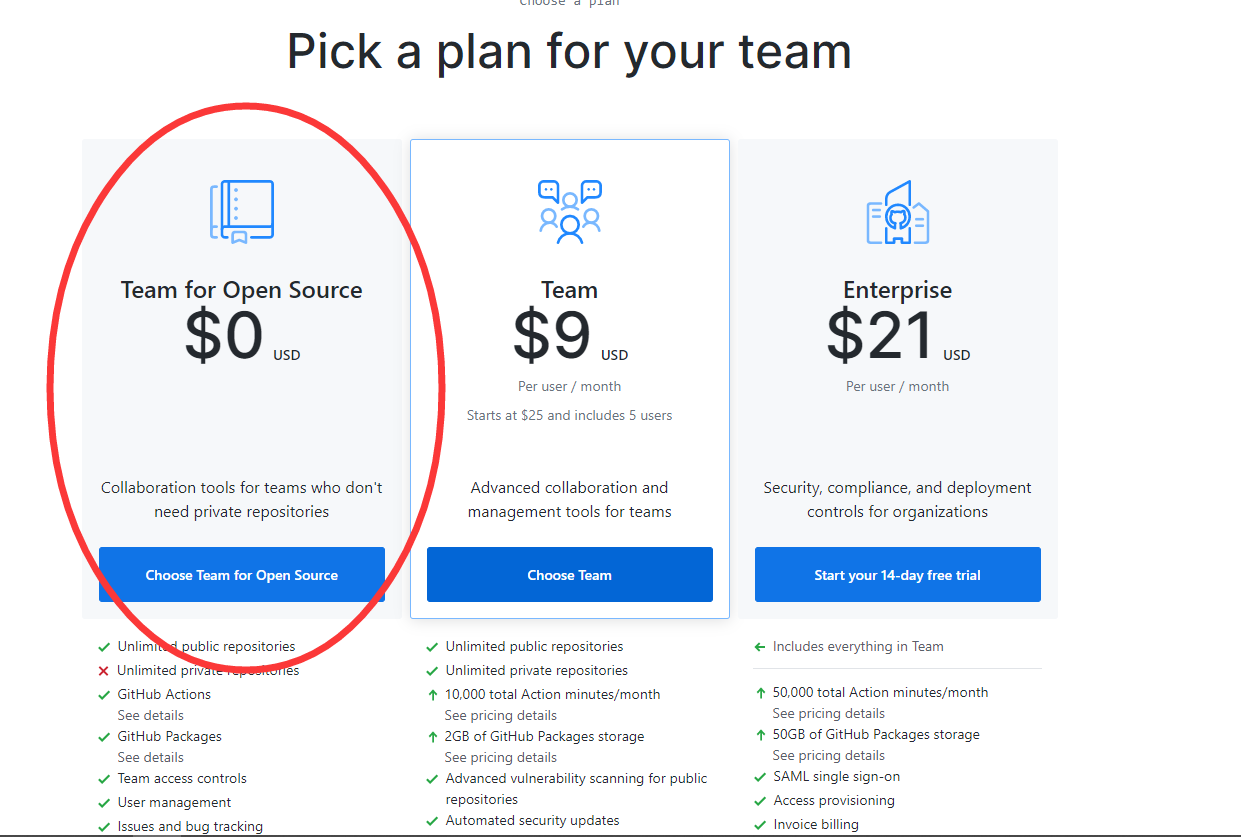


图2-55 选择创建新组织

第二步，邀请团队成员这一步我们可以通过他们的github昵称邀请团队成员，也可以稍后在项目主页中进行邀请，如图2-56所示。

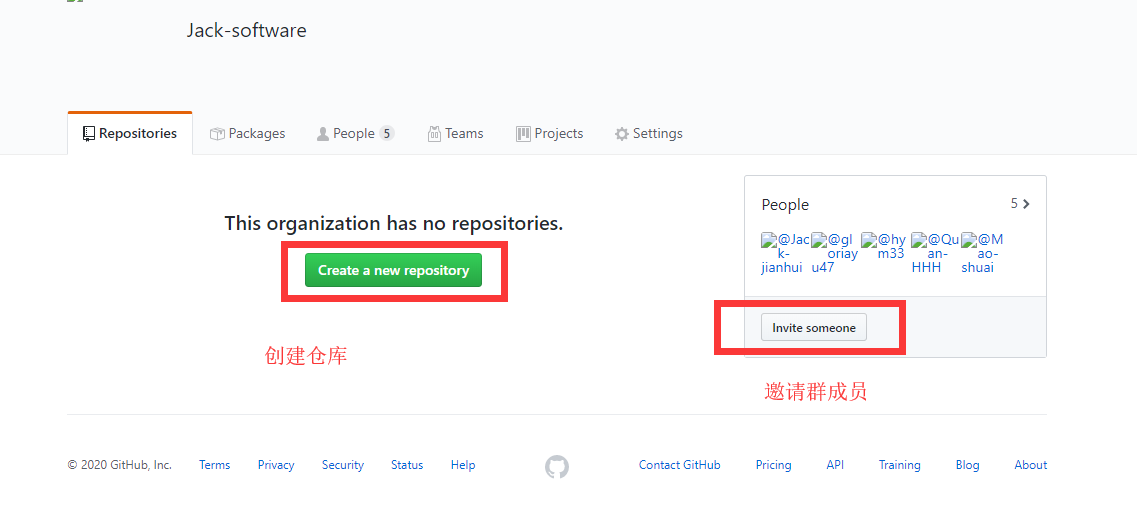


图2-56 邀请组织成员

接下来我们首先需要在图 4 菜单栏中选择 Settings，当每创建的一个新组织时处创建者外其余人员的默认权限是没有 push 权限（Read）的，需要手动改为 Write 或 Admin 权限，如图2-57所示。

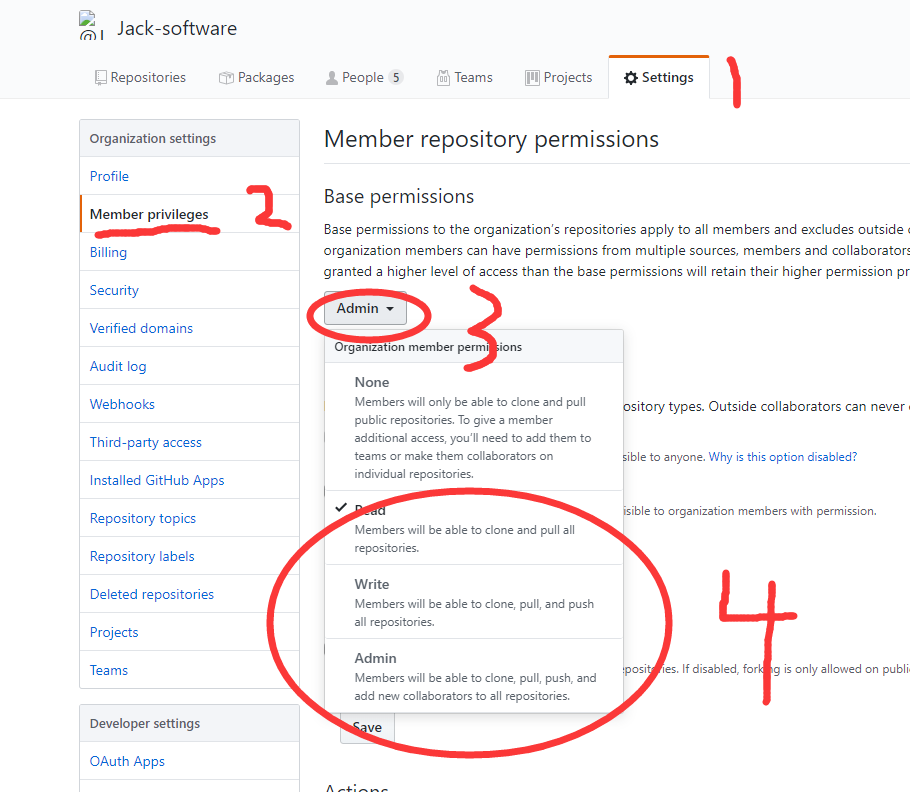


图2-57 更改组织成员权限

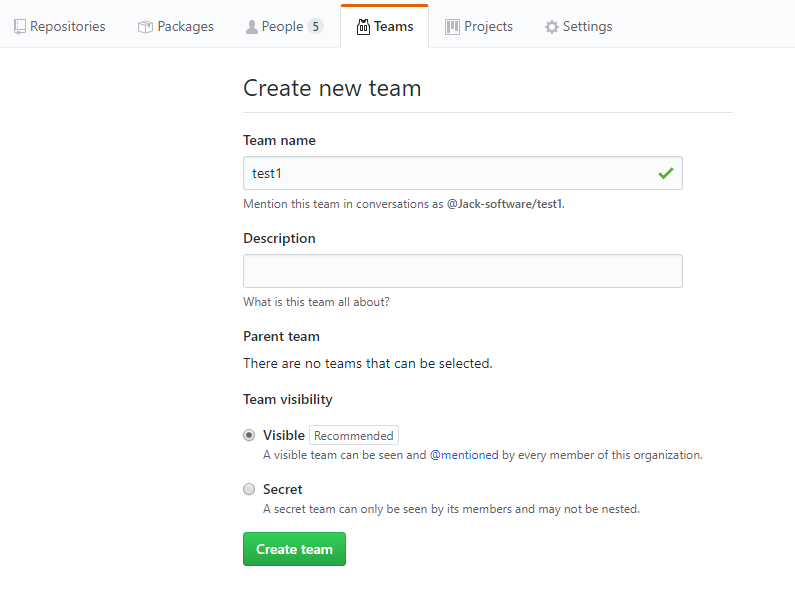
我们可以选择到team 页面创建属于自己的团队，在GitHub中，一个组织可以分为多个小组，小组允许您与组内的其他人员共享设置权限，不同的组有不同的权限分配，可以极大的提高各分组的效率，下面我们可以来创建我们的开发团队，如图2-58和图2-59 所示。 

图2-58 创建团队

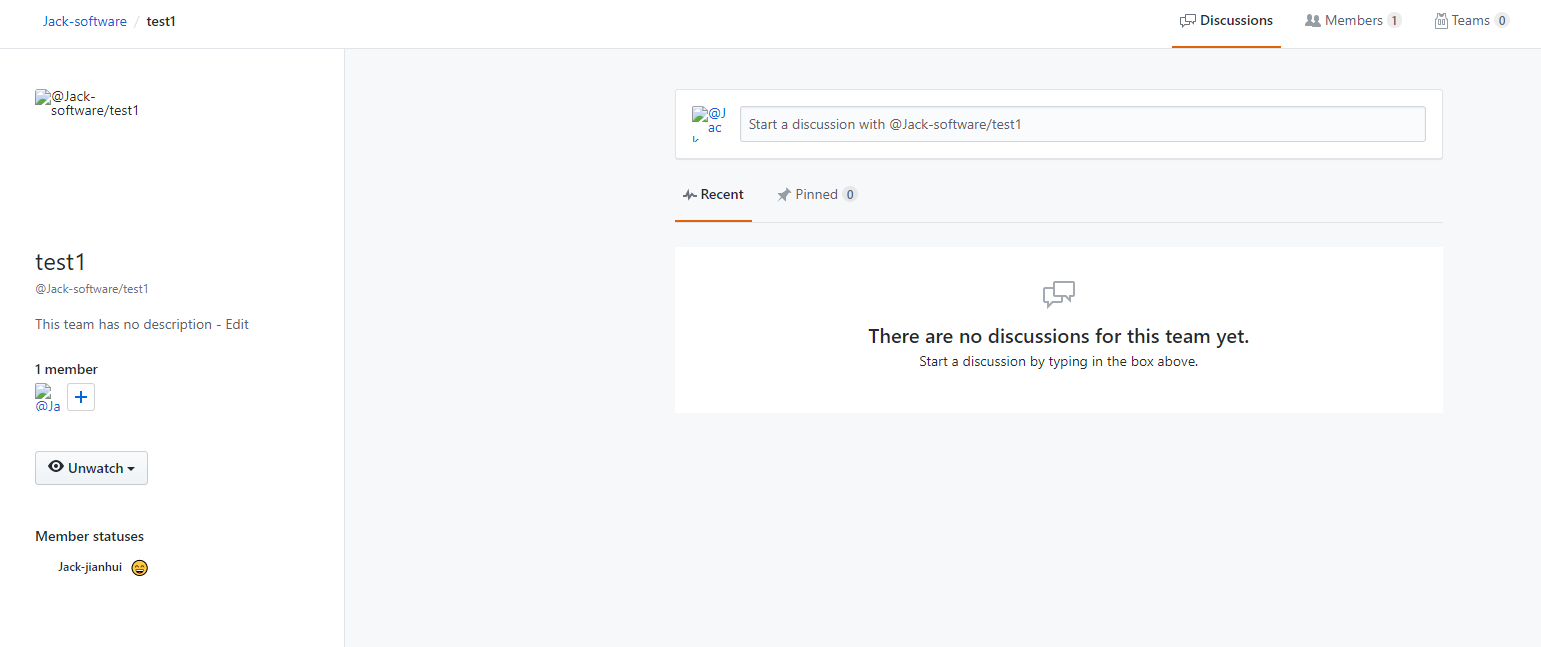


图2-59 团队界面

返回到首页，现在我们可以选择创建一个新的仓库来存储你本地仓的文件，如图 2-60所示。

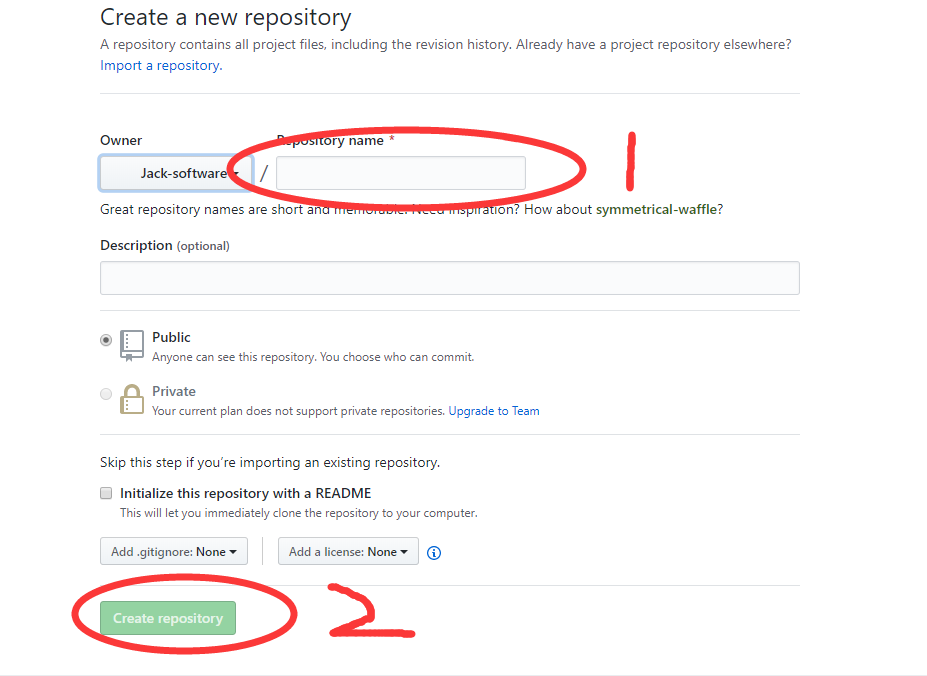


图2-60 创建仓库

建立好仓库后会进入到repository界面，如图2-61所示。

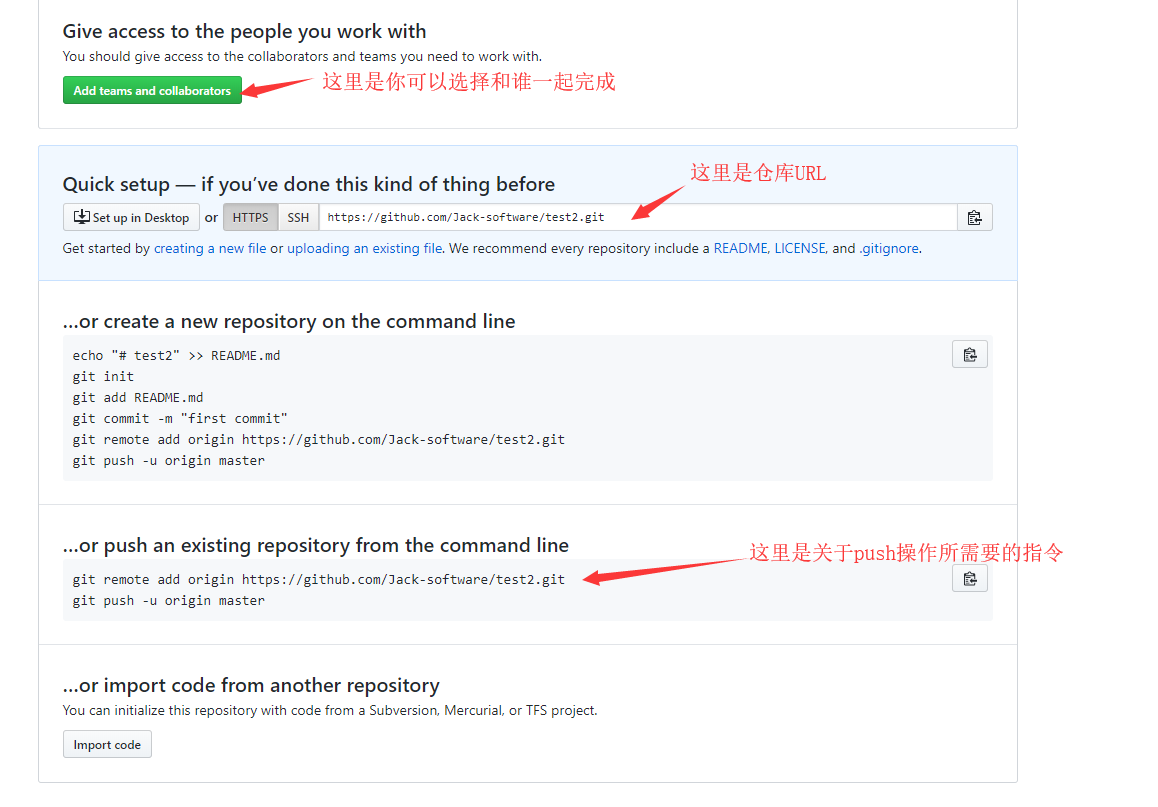


图2-61 repository初始界面

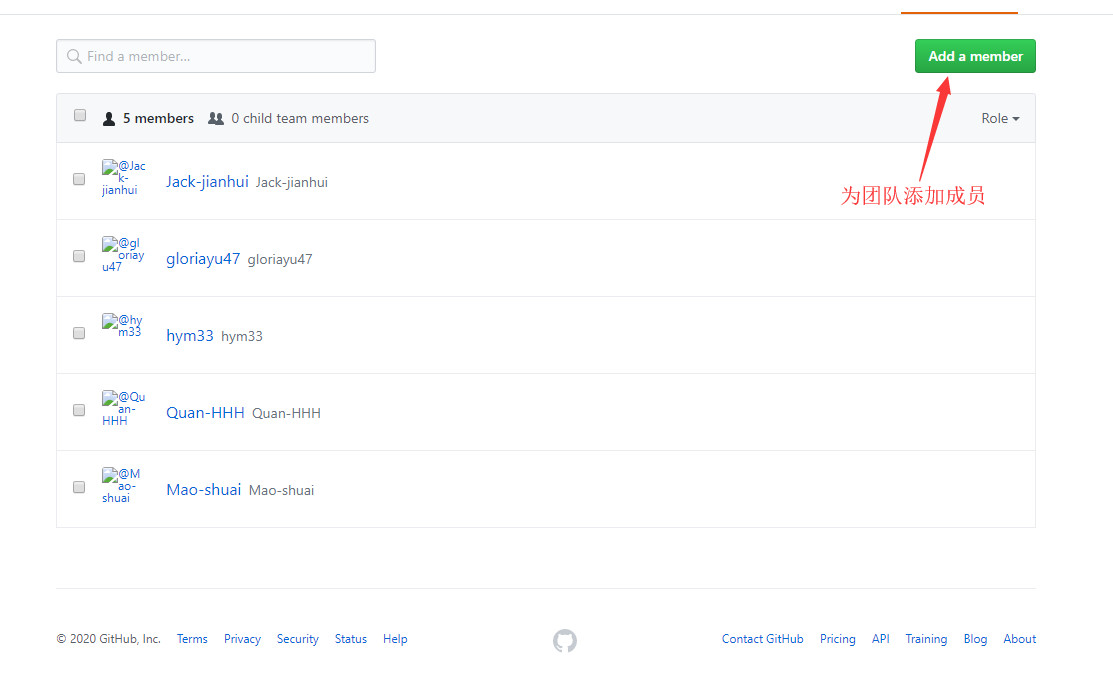
接下来我们可以点击Add teams and collaborators 按钮，进入到选择添加成员页面，并将上面已经创建好的 team 加入到本仓库中，并可对不同的项目进行权限划分。如图2-62、图 2-63 所示和图 2-64 所示。 

图2-62 为团队添加成员

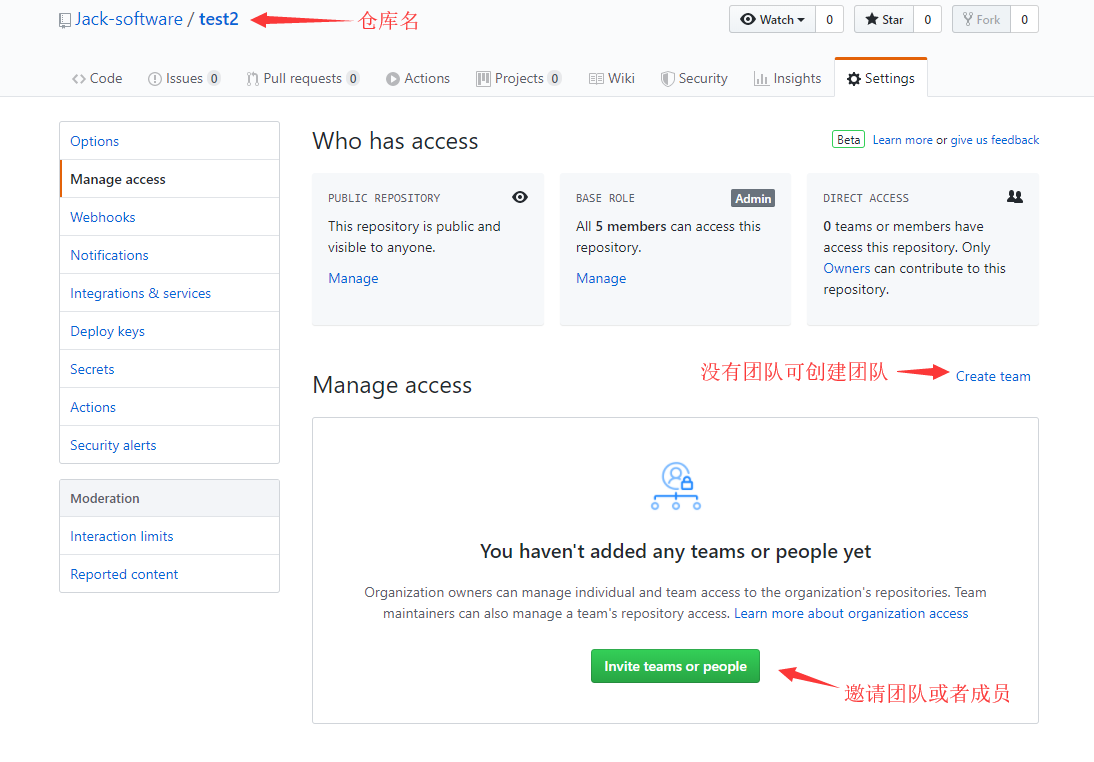


图2-63 为仓库添加团队

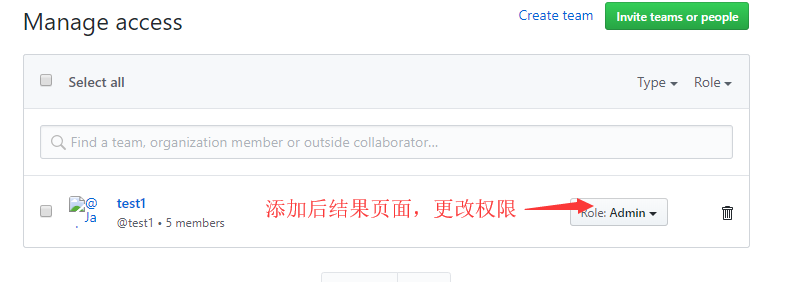


图2-64 更改团队权限

# 3 组员操作git记录

在多人协作中，每个组员都将已建立好的一个远程仓库克隆到自己的文件夹中，并自己再创建一个文件，在上传到远程仓库中。

詹健辉：克隆远程仓库如图3-1所示。上传自己的文件，并上传到远程仓库。如图3-2所示。

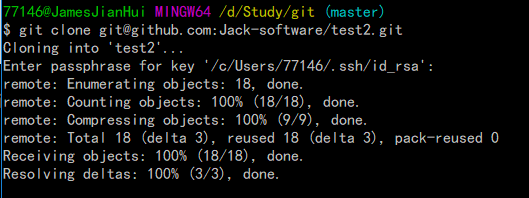


图3-1 克隆远程仓库

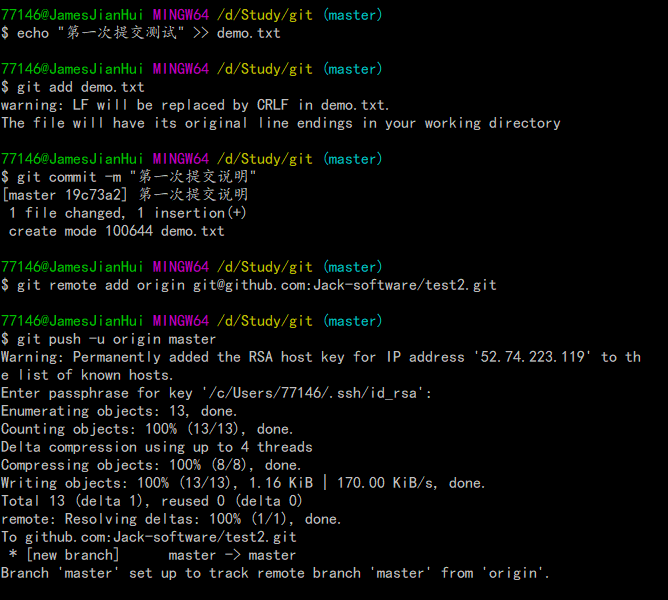


图3-2 将本地仓库上传到远程仓库

权佳欣：克隆远程仓库并上传自己的文件到远程仓库，如图3-3所示。

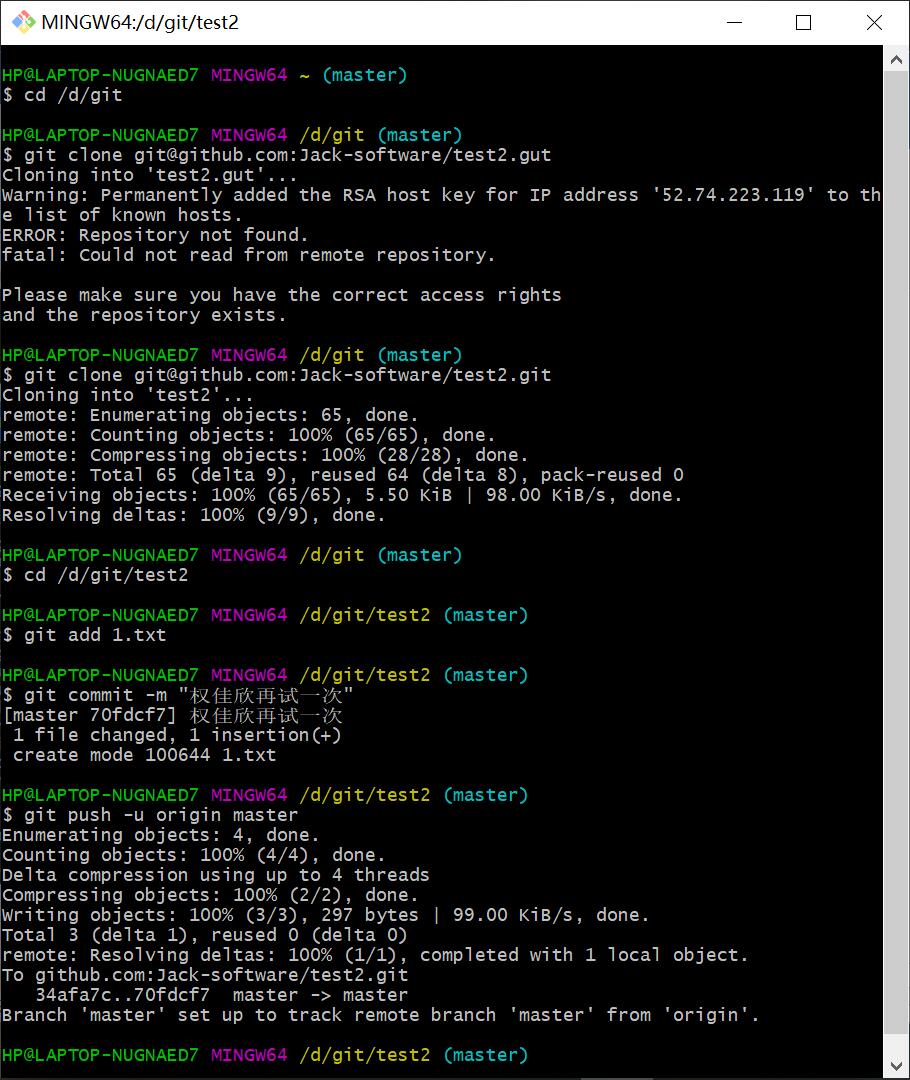


图3-3 克隆远程仓库并上传自己的文件到远程仓库

黄友民：克隆远程仓库如图3-4所示。上传自己的文件，并上传到远程仓库。如图3-5所示。

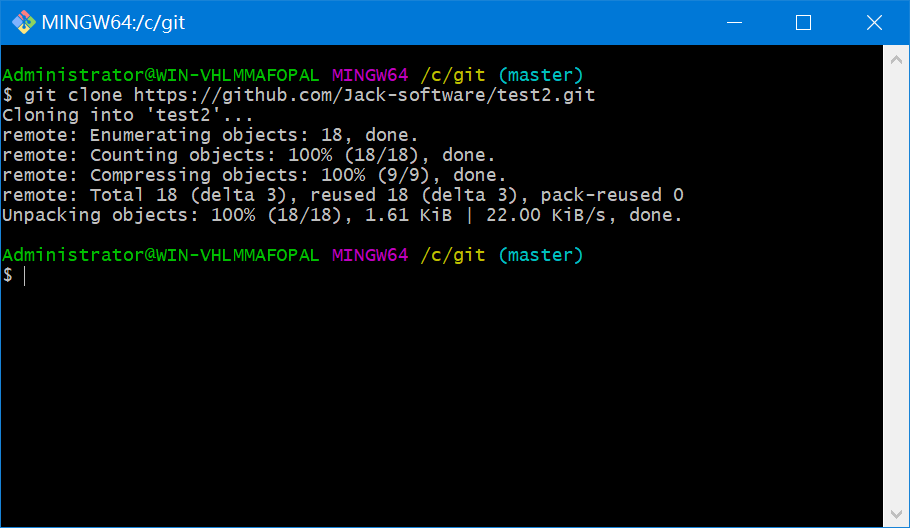


图3-4 克隆远程仓库

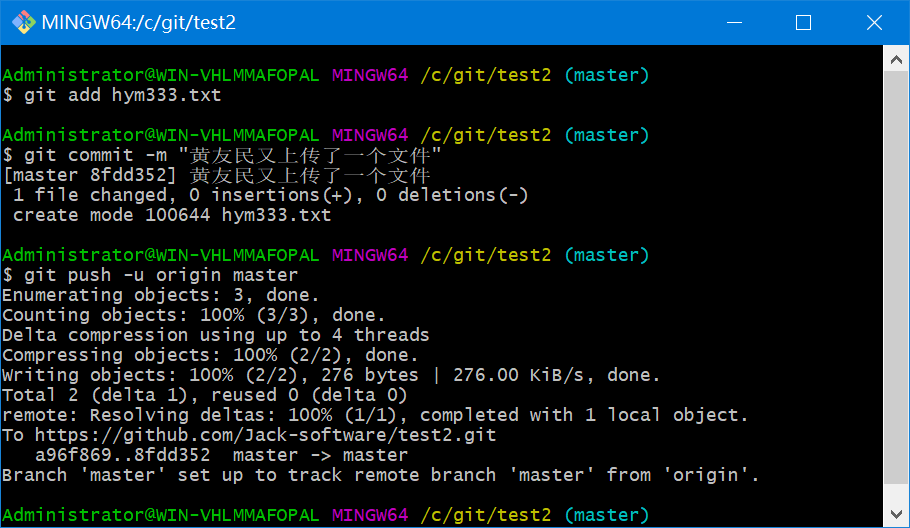


图3-5 将本地仓库上传到远程仓库