|  |  |  |  |
| --- | --- | --- | --- |
| 作者： | 刘蒙 | | [Logo_Beckhoff_Red](http://www.beckhoff.com.cn/)  中国上海市静安区汶水路 299 弄 9-10 号  市北智汇园4号楼（200072）  TEL: 021-66312666  FAX: 021-66315696 |
| 职务： | 风电部 | 技术工程师 |
| 日期： | 2020-5-25 | |
| 邮箱： | Meng.liu @beckhoff.com.cn | |
| 电话： |  | |

|  |
| --- |
| **在TwinCAT3中使用Git进行版本管理** |
| **摘 要**：本文介绍了分布式版本控制系统Git的安装及基本命令，如何在TwinCAT3工程中使用Git进行版本管理，以及将Git与Github仓库关联使用。 |
| **关键字：**TwinCAT 3，Git，Github，版本管理 |
| **附 件：**   |  |  |  | | --- | --- | --- | | 序 号 | 文件名 | 备注 | |  |  |  | |  |  |  | |  |  |  | |  |  |  | |
| **历史版本：**   |  |  |  | | --- | --- | --- | |  |  |  | |  |  |  | |  |  |  | |  |  |  | |  |  |  | |
| **免责声明：**  我们已对本文档描述的内容做测试。但是差错在所难免，无法保证绝对正确并完全满足您的使用需求。本文档的内容可能随时更新，也欢迎您提出改进建议。 |
| **参考信息：** |

**目 录**

目录

[2. 概述 3](#_Toc41297278)

[2.1. 什么是Git？ 3](#_Toc41297279)

[2.2. TwinCAT3 + Git 能解决什么问题？ 3](#_Toc41297280)

[3. 使用Git管理TwinCAT3工程（基础） 4](#_Toc41297281)

[3.1. Git的安装 4](#_Toc41297282)

[3.2. 使用Git常用命令管理TwinCAT3工程 5](#_Toc41297283)

[4. 使用Git管理TwinCAT3工程（进阶） 12](#_Toc41297284)

[4.1. .gitignore文件 12](#_Toc41297285)

[4.2. Git分支管理 13](#_Toc41297286)

[4.3. 使用Git图形化操作工具gitk 15](#_Toc41297287)

[4.4. 在VScode中使用git 16](#_Toc41297288)

[5. 连接远程仓库Github 17](#_Toc41297289)

[5.1. Github概述 17](#_Toc41297290)

[5.2. 连接到远程仓库 17](#_Toc41297291)

[5.3. 把本地工程上传到Github 18](#_Toc41297292)

[6. Git命令速查表 23](#_Toc41297293)

# 概述

## 什么是Git？

Git是一个开源的分布式版本控制系统软件，可以有效、高速地处理从很小到非常大的项目版本管理。

版本控制是一种用于记录一个或多个文件内容变化，方便查阅特定版本修订情况的系统。以前在没有使用版本控制的时候，通常在项目根目录下这样命名项目：project\_v1、project\_v1.1、project\_v2等等，通过这种方式记录项目不同版本的修改，有的时候我们还会在不同版本的文件中写一个说明，记录此版本项目新增、修改，删除等操作。这样的操作是很繁杂的，有的时候还可能因为一些非人为因素导致文件丢失这样的事故。有了版本控制系统，我们就不用再手动进行一些繁杂的操作，并且对于文件丢失这种事故也不用再担心，可以随便回到历史记录的某个时刻。

和集中式版本控制系统SVN相比，由于Git是分布式的系统，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，比集中式版本控制系统更能提高版本管理效率。

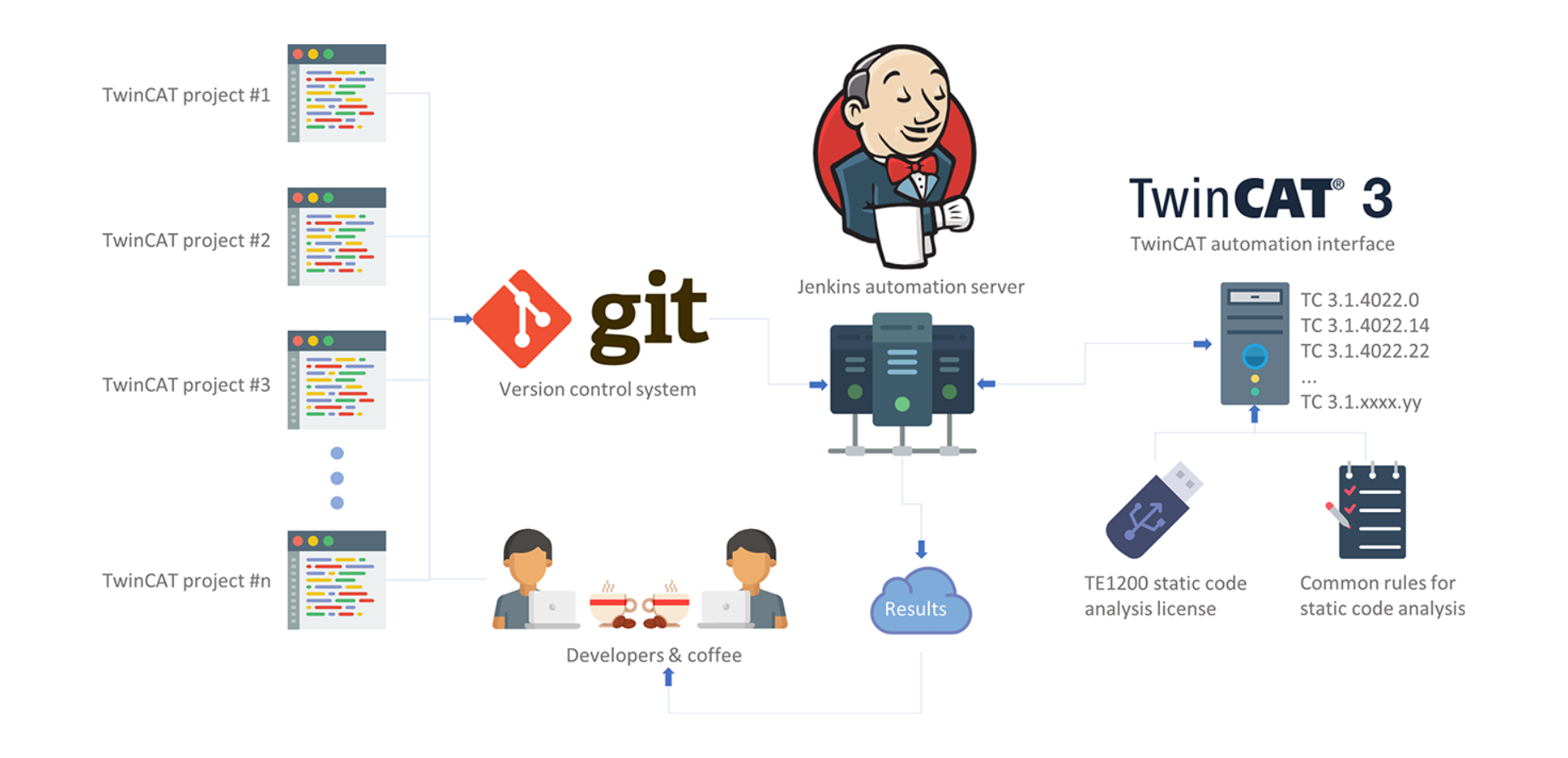
GitHub是一个基于 Git的用于远程代码存储和版本管理的网站，既为开发团队提供了存储、分享、发布和合作开发项目的中心化云存储的场所，也提供了分享开源项目的平台。

## TwinCAT3 + Git 能解决什么问题？

当使用TwinCAT3进行软件开发时，通过使用Git可以实现多人之间的协作开发，包括版本管理、分支合并、版本回退等功能，提高开发效率。

当使用TwinCAT3进行现场调试时，通过使用Git可以保存并记录每个调试阶段的代码内容，可以随时回退版本，查看之前调试的代码情况。

甚至，在TwinCAT3/Git/Github的基础上，结合Jenkins和automation interface的使用，实现代码的持续集成和持续发布（CICD），以及整个开发运维（DevOps）的全自动化流程。这样当开发工程师更新了TwinCAT3程序内容后，只需要一键上传到Github仓库，即可通过Jenkins和automation interface，自动完成程序的编译，激活以及下载到目标控制器，可以快速的完成工业现场几十甚至上百台倍福控制器的程序批量下载更新，提高现场运维的效率。

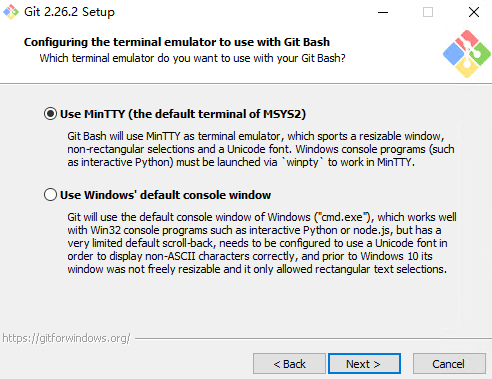
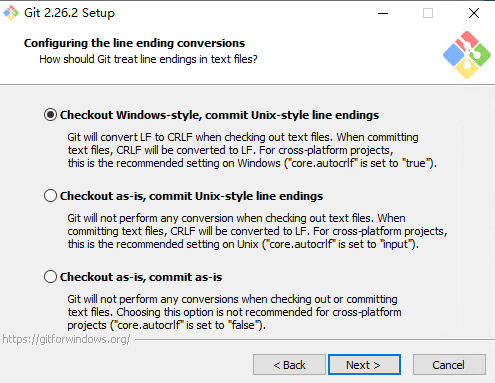
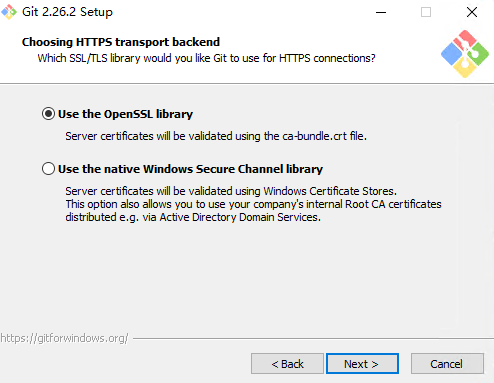
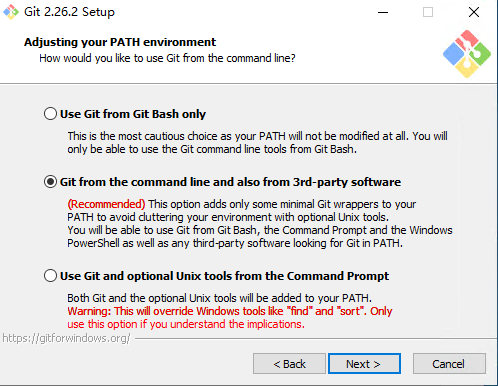


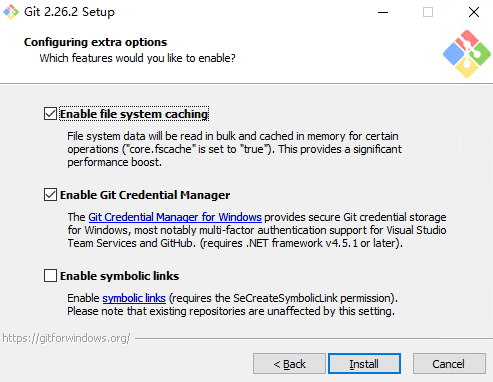
# 使用Git管理TwinCAT3工程（基础）

## Git的安装

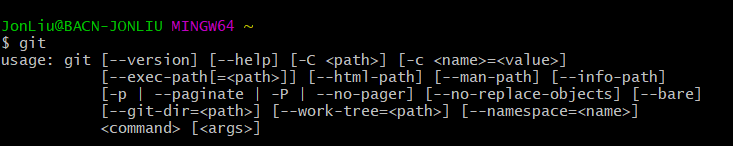
在网站<https://git-scm.com/download/win>下载windows版Git软件安装包，当前版本为Git-2.26.2-64-bit.exe。

双击安装包进行安装，以下选项全部选择默认即可。





安装完成后，通过cmd，Powershell以及开始菜单中的Git bash，都可以用命令行的方式使用Git。在Git bash中输入git，可以看到弹出了Git的帮助信息，证明Git安装成功了。



## 使用Git常用命令管理TwinCAT3工程

当电脑安装完Git软件后，就可以在电脑中创建版本库。

什么是版本库呢？版本库又名仓库，英文名repository，可以简单理解成一个目录，这个目录里面的所有文件都可以被Git管理起来，每个文件的修改、删除，Git都能跟踪，以便任何时刻都可以追踪历史，或者在将来某个时刻可以“还原”。

也就是说，在一台电脑上可以有多个版本库，一个版本库对应一个文件路径，一个版本库可以管理一个工程项目。后面内容中使用仓库或者版本库，都是指的一个概念。

了解了这些内容后，我们通过命令git init创建仓库。

1. git init

先在桌面新建一个文件夹myproject，打开vs创建一个新的TwinCAT Project1存放在myproject文件夹中。在..\Desktop\myproject\TwinCAT Project1\目录下，鼠标右键空白处，点击Git bash here，弹出下图git bash黑色对话框。在对话框中输入git init命令，即可把当前文件目录变成一个git仓库。

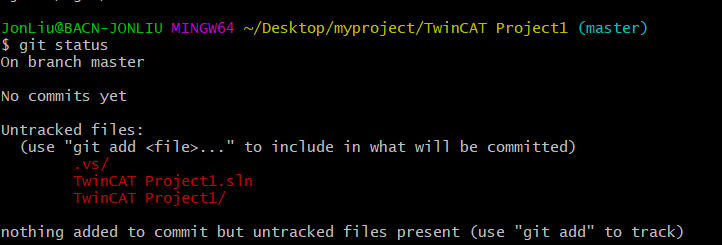


完成上述步骤后，可以看到在文件夹中多了一个.git/文件夹，该文件夹就用来存放仓库的信息。请不要删除或更改里面的内容。

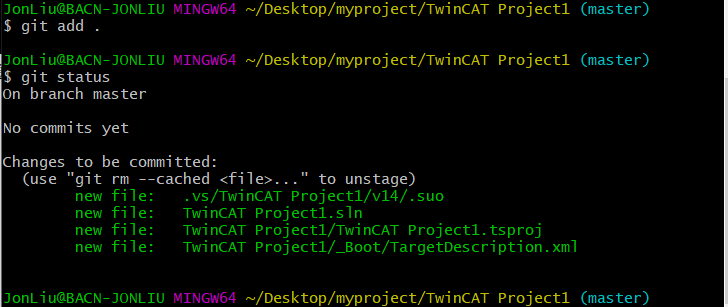


1. git status / git add / git commit

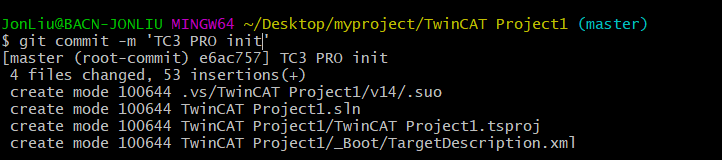
git status可以让我们时刻掌握仓库当前的状态，使用git status命令，git会输出当前版本库的状态信息，下图git告诉我们版本库里有几个文件没有被追踪（untracked）。这也是新建了TC3工程后的初始文件内容。



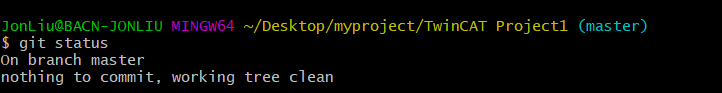
按照git status的提示，通过git add可以把文件添加到仓库。使用git add .命令，把当前目录下的所有文件全部添加进仓库。然后使用git status命令查看当前仓库的状态，可以看到git告诉我们，有new file被追踪了，还没有提交到仓库。



按照git status的提示，使用git commit -m 'TC3 PRO init'命令，把文件提交到仓库。-m ‘TC3 PRO init’是指给这次提交添加的版本提交描述信息。显示如下：



使用git status命令查看当前仓库的状态，我们可以看到，Git告诉我们当前没有需要提交的修改，而且，工作目录是干净（working tree clean）的。



通过以上步骤，我们就完成了一次版本提交。

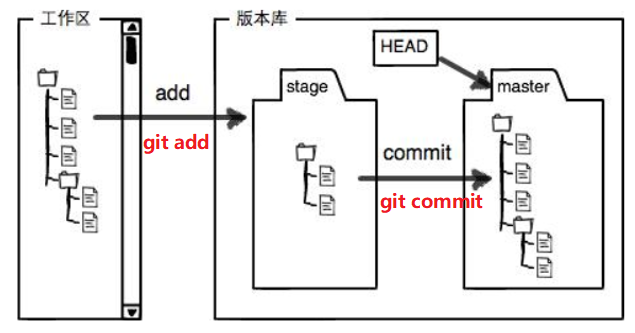
1. 工作区和暂存区的概念

Git和其他版本控制系统如SVN的一个不同之处就是有暂存区的概念。

工作区就是你在电脑里能看到的目录，比如我们桌面的myproject\TwinCAT Project1\文件夹就是一个工作区。

工作区有一个隐藏目录.git，这个不算工作区，而是Git的版本库。

Git的版本库里存了很多东西，其中最重要的就是称为stage（或者叫index）的暂存区，还有Git为我们自动创建的第一个分支master，以及指向master的一个指针叫HEAD。



在b小节讲了把文件往Git版本库里添加的时候，是分两步执行的：

第一步是用git add把文件添加进去，实际上就是把文件修改添加到暂存区；

第二步是用git commit提交更改，实际上就是把暂存区的所有内容提交到当前分支。

有了暂存区，可以多次git add，然后再执行git commit。

简单来说，当我们修改了程序后，想把当前的程序提交到仓库，就先后执行git add和git commit两个命令即可。

1. 添加PLC程序

接下来我们打开TC3工程，如下图：

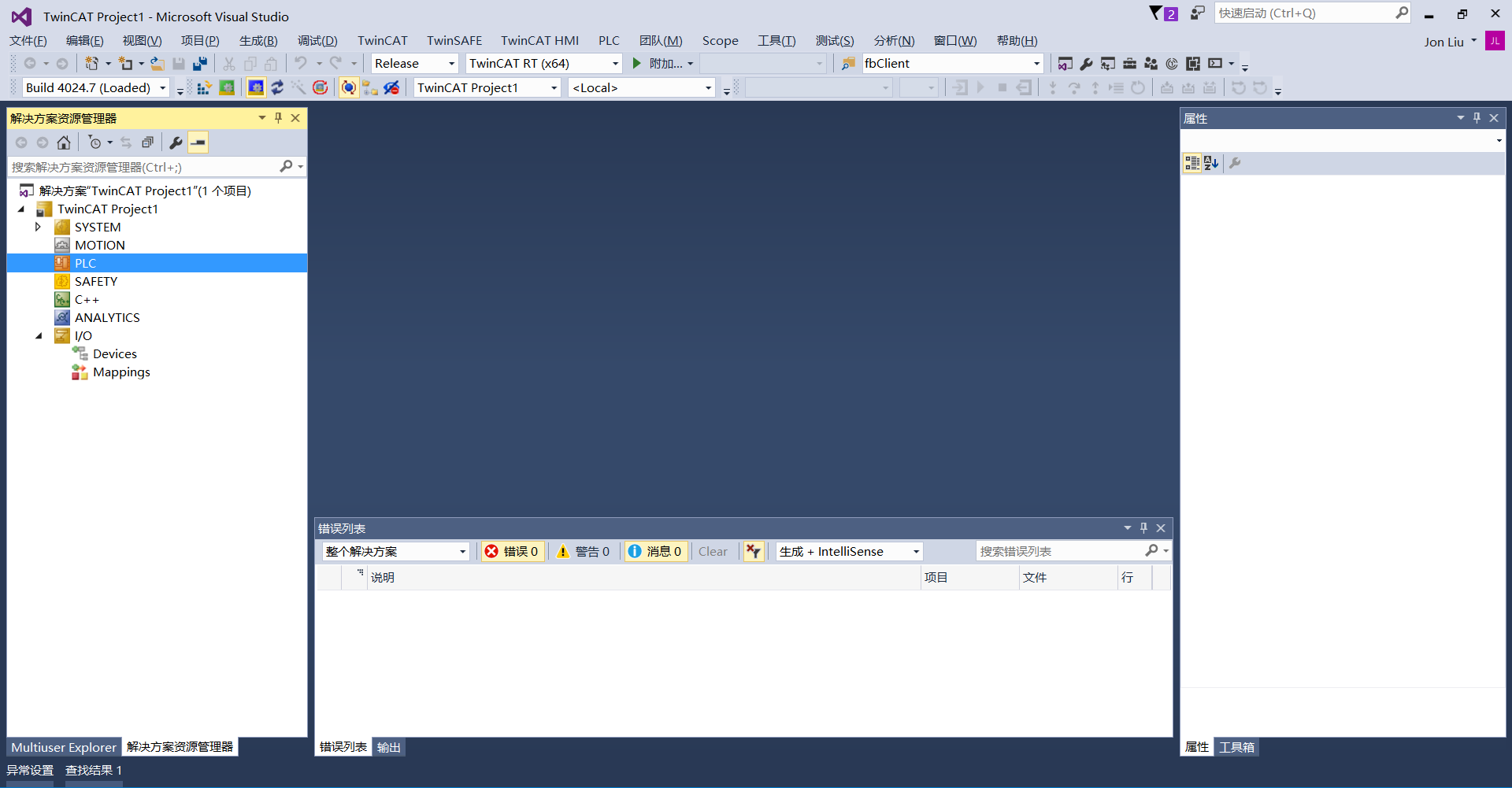
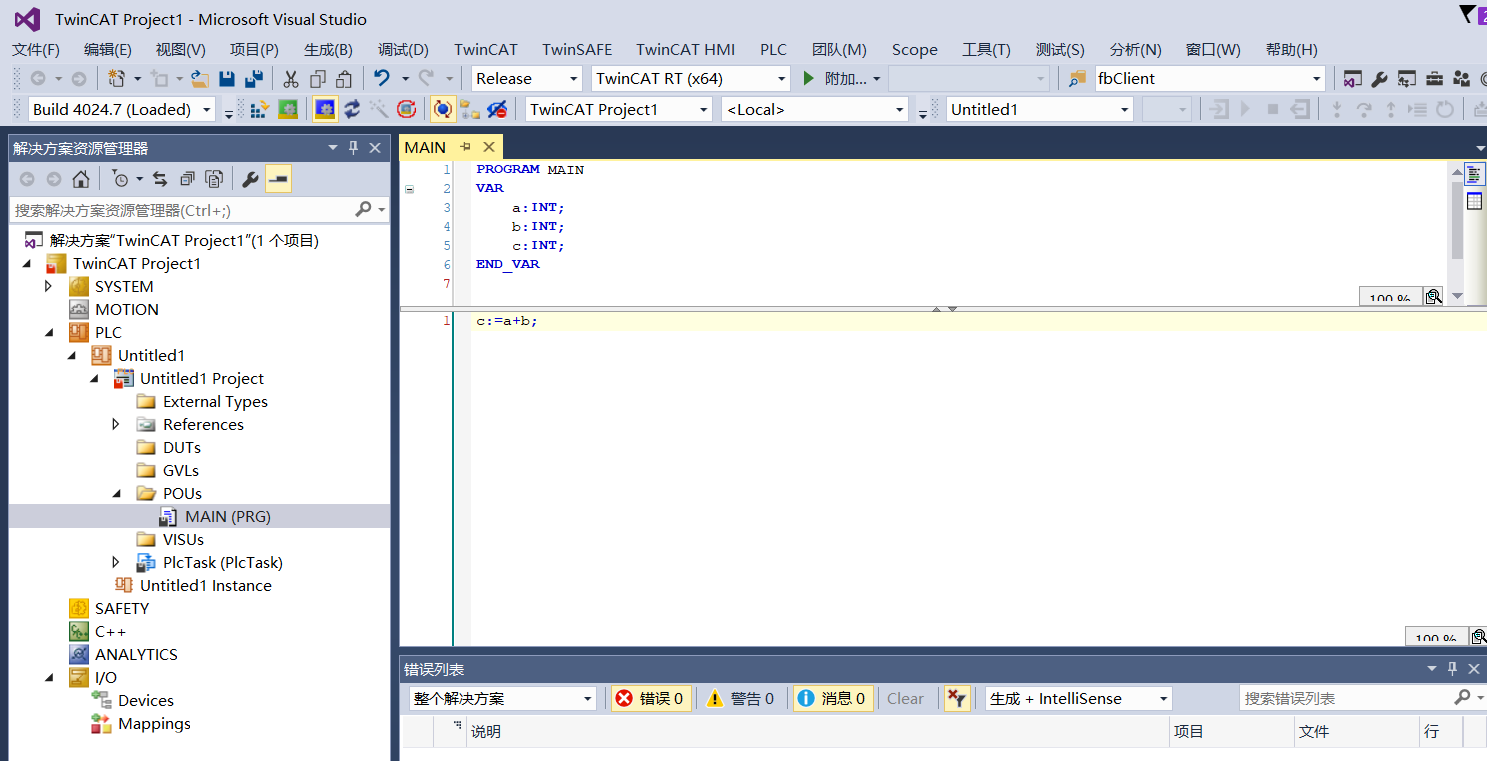


图 d1

可以看到是一个全新的工程，没有添加PLC和I/O。这也是在版本库中第一次提交的的状态。

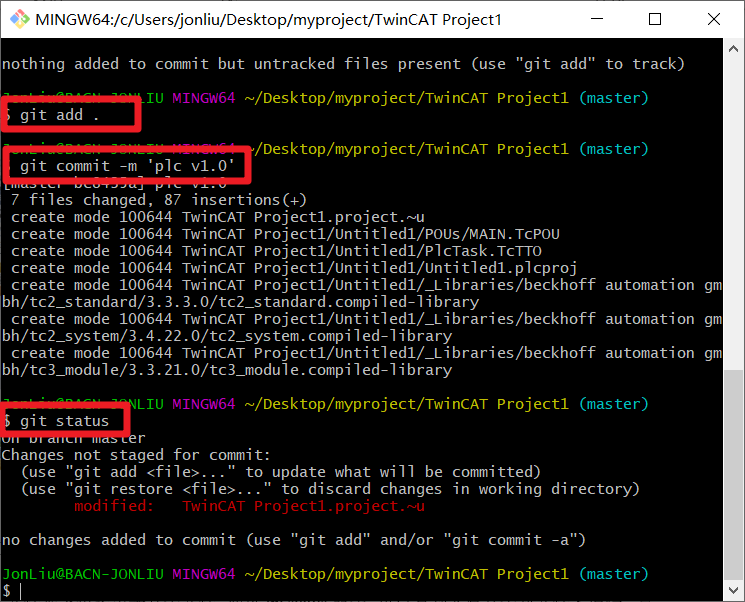
接下来我们新添加一个PLC工程，并在main中添加如下代码，并保存程序：



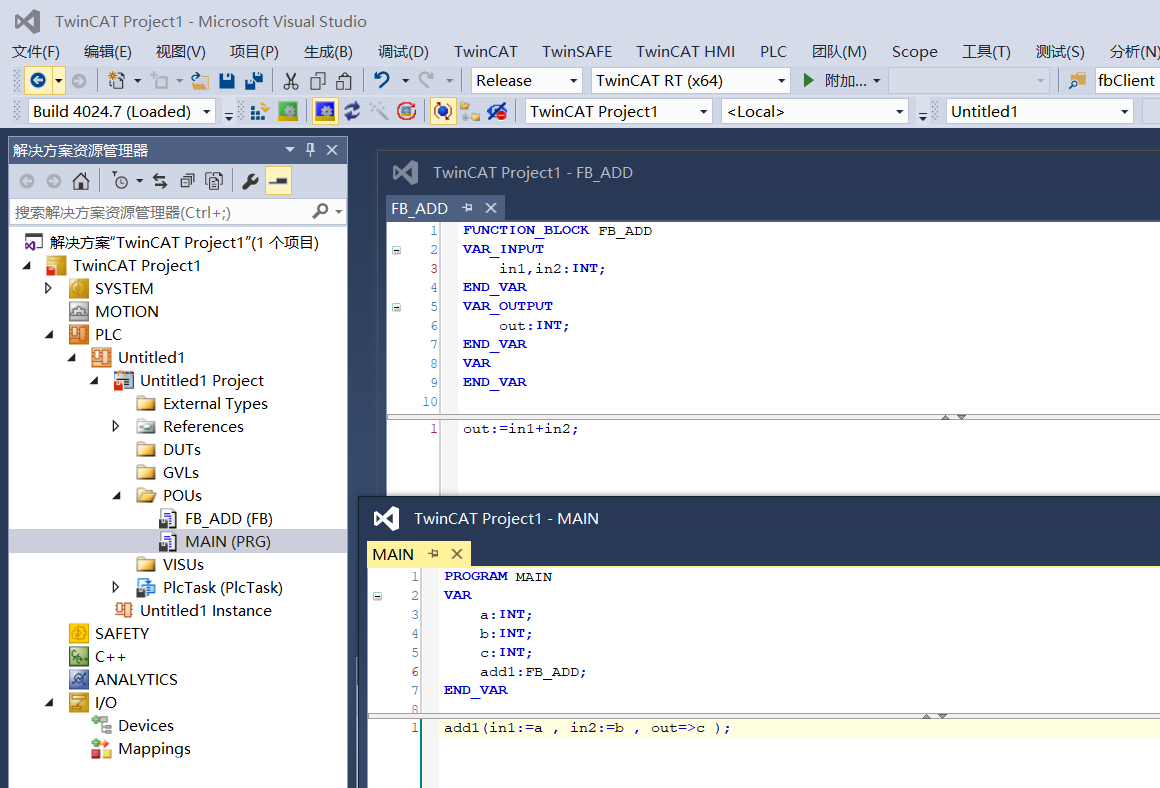
图d2

执行下图中红框的三个命令，完成第二次版本的提交，提交的版本描述信息为plc v1.0。

这次我们执行完add和commit后，并没有显示工作目录是干净（working tree clean）的。这是因为TwinCAT Project1.project.~u文件一直在自动更新的缘故。暂时先不用管他，后面会通过.gitignore文件把不需要进行版本管理的文件剔除掉。



接下来创建一个FB\_ADD，并修改main中的内容，然后保存。内容如下：

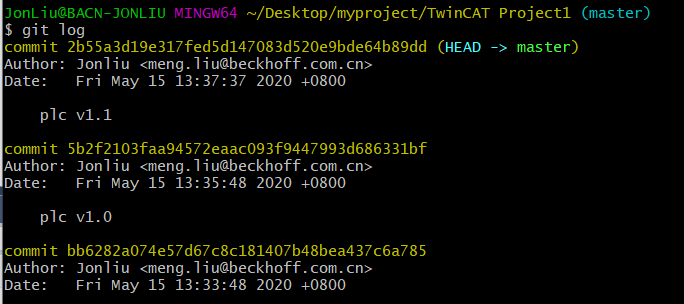


此时进行第三次提交，执行的命令分别为

git add .和git commit -m 'plc v1.1'

1. git log / git reflog

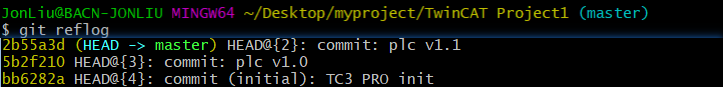
目前git仓库里有了3次commit的版本，可以通过git log命令，查看所有提交过的版本信息。



git log可以显示所有提交过的版本信息，不包括已经被删除的 commit 记录和 reset 的操作。

git reflog是显示所有的操作记录，包括提交，回退的操作。一般用来找出操作记录中的版本号，进行回退。git reflog常用于恢复本地的错误操作。

git reflog显示的内容如下：



1. 版本回退git reset --hard head^ / git reset --hard <id>

如果此时，想把程序回退到plc v1.0。

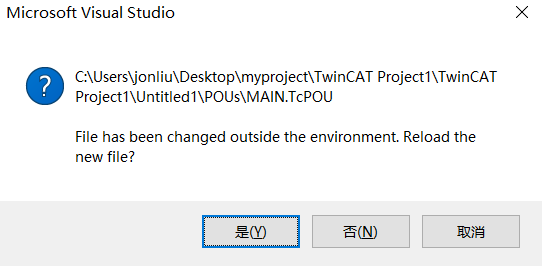
方法一，可以执行git reset –-hard head^命令。



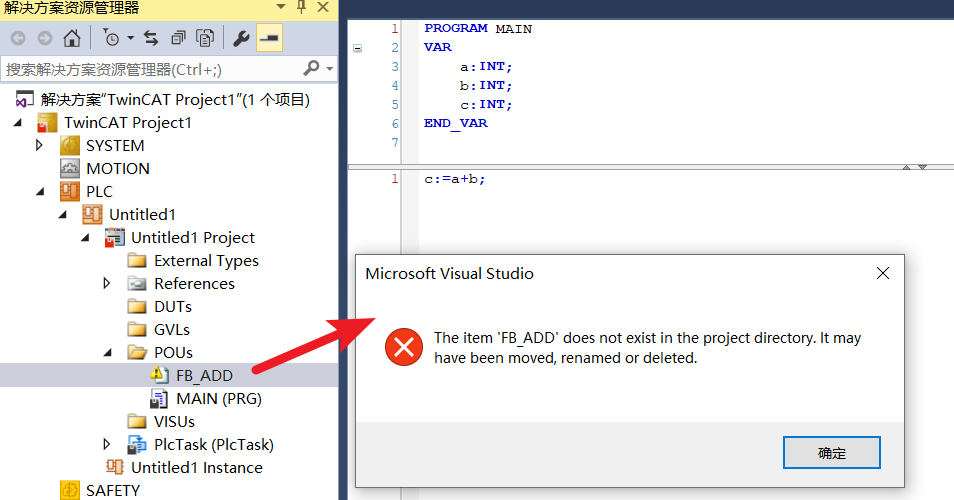
在Git中，用HEAD表示当前版本，也就是最新的提交0afa1c4，上一个版本就是HEAD^，上上一个版本就是HEAD^^，当然往上100个版本写100个^比较容易数不过来，所以写成HEAD~100。

方法二，可以执行git reset -–hard 5b2f210命令。版本号没必要写全，前几位就可以了，Git会自动去找。当然也不能只写前一两位，因为Git可能会找到多个版本号，就无法确定是哪一个了。

执行以上任意命令后，VS会弹出对话框，点击是

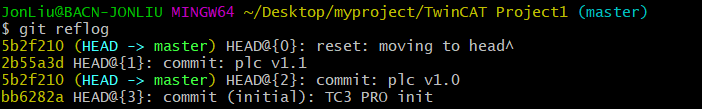


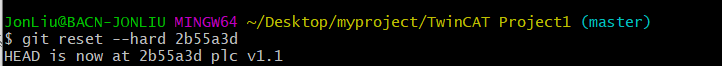
当点击FB\_ADD功能块时，会弹出提示框，文件已被删除。



如果此时，又想把程序恢复到plc v1.1。

通过命令git reflog找到plc v1.1的commit id，执行git reset –-hard 2b55a3d命令即可。

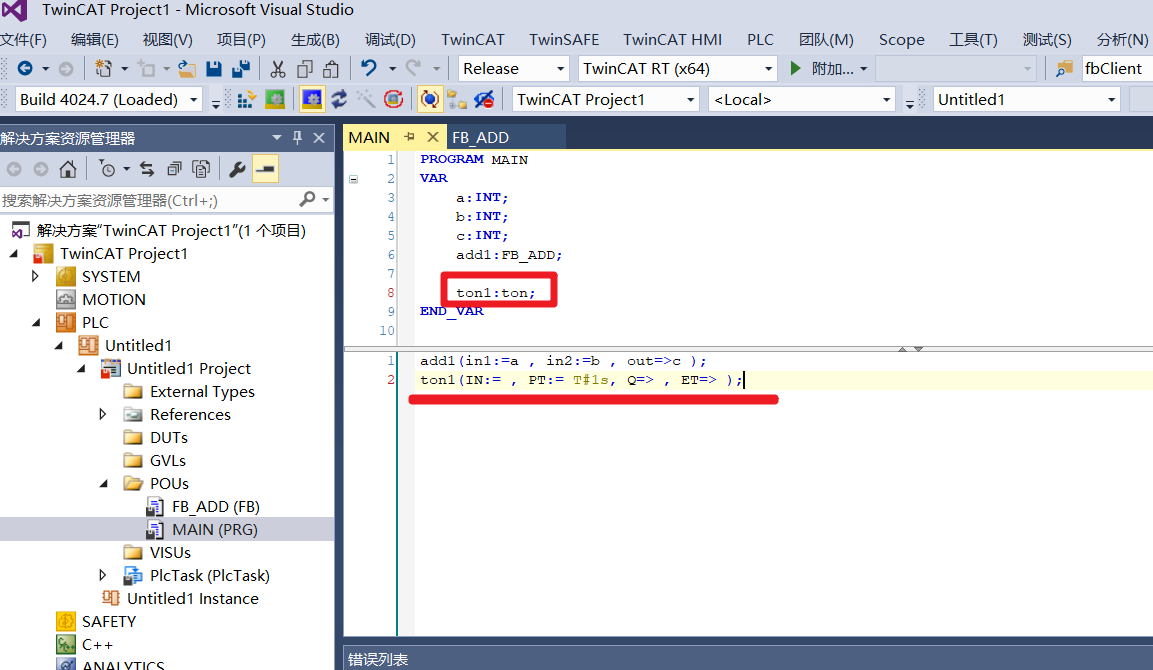




可以看到TC3中，FB\_ADD功能块又恢复回来了。

1. 丢弃工作区中的修改git restore .

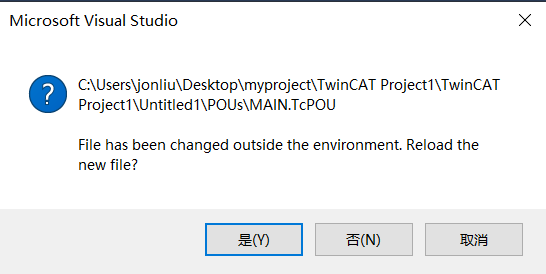
接下来在main中添加一部分内容。内容如下：



此时，如果我想丢弃main中新添加的这部分代码，则执行git restore .命令。



执行命令后，vs弹框如下，点击是。



可以看到main里面修改的内容回退到版本库里的内容了。

# 使用Git管理TwinCAT3工程（进阶）

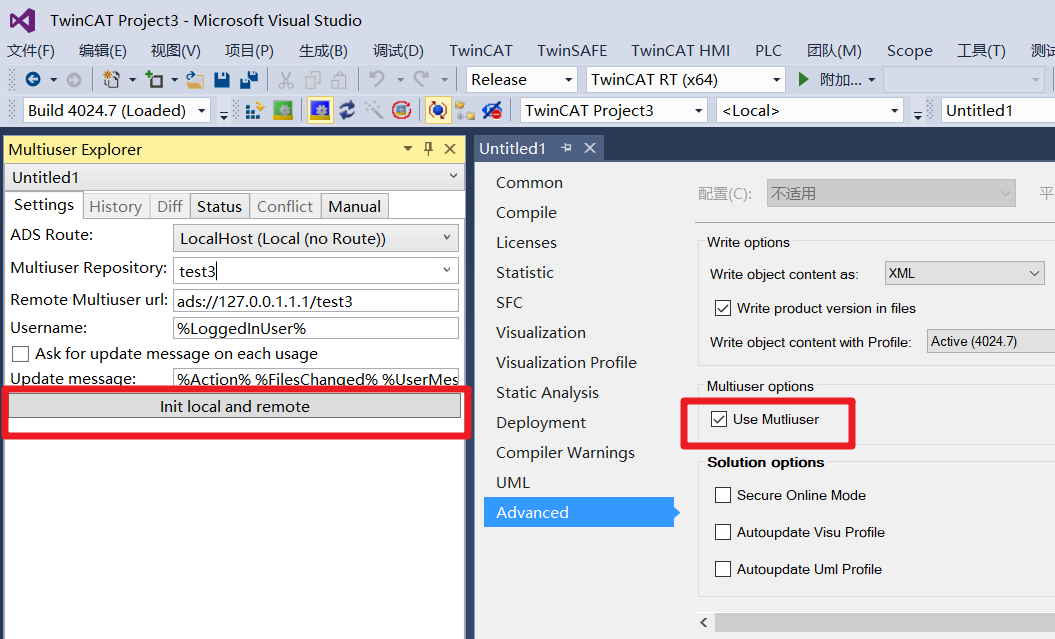
## .gitignore文件

每个Git仓库中都需要一个“.gitignore”文件，这个文件的作用就是告诉Git哪些文件不需要添加到版本管理中。在TwinCAT3工程中有很多在编译和激活过程中自动生成的文件，以及一些备份文件。我们通过“.gitignore”文件把这些文件都忽略掉。

1. 使用倍福官方提供的“.gitignore”文件

TC3 4024版本的multi-user提供了倍福官方的PLC工程的“.gitignore”文件，可以直接使用它来管理PLC project的版本。

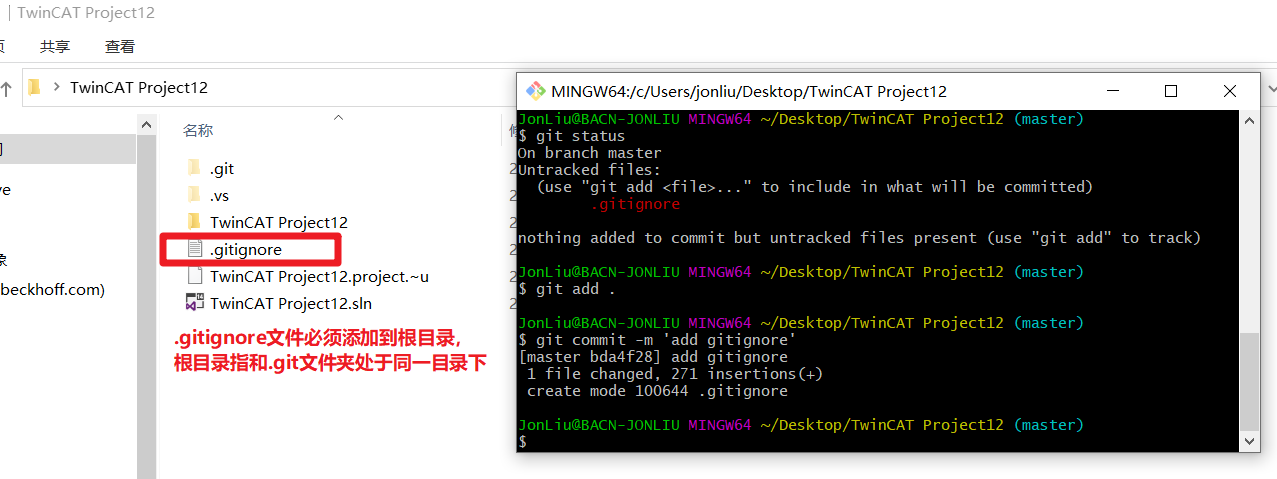
新建一个TC3工程，创建PLC project，按照下图，选中user Multiuser，在multiuser explorer窗口填写相关内容，点击init local and remote。



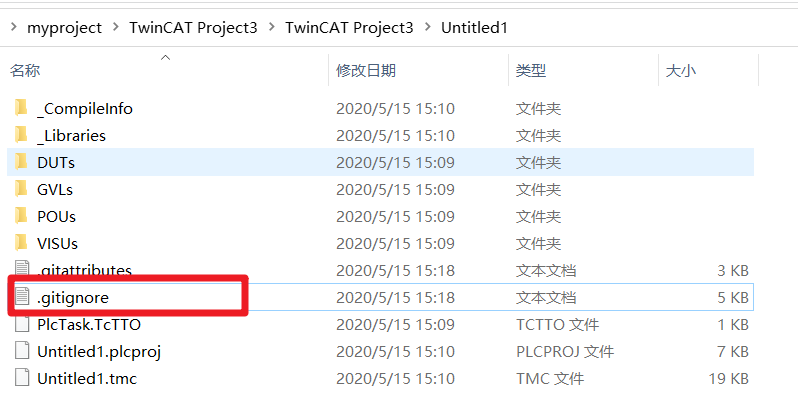
成功后，在文件夹中出现了“.gitignore”文件。

注意，我们通过此步骤仅为获取“.gitignore”文件，使用Git管理TwinCAT3项目时，不要在TC3中开启mutliuser，两者会产生冲突。

以后新建git仓库后，把该文件拷贝到仓库根目录下即可。然后通过执行git add .和git commit -m ‘add gitignore’ ，把“.gitignore”文件添加到版本管理中即可。



如果想用Git管理TwinCAT HMI工程，不需要拷贝该文件。新创建的TwinCAT HMI工程中会自动生成“.gitignore”文件。



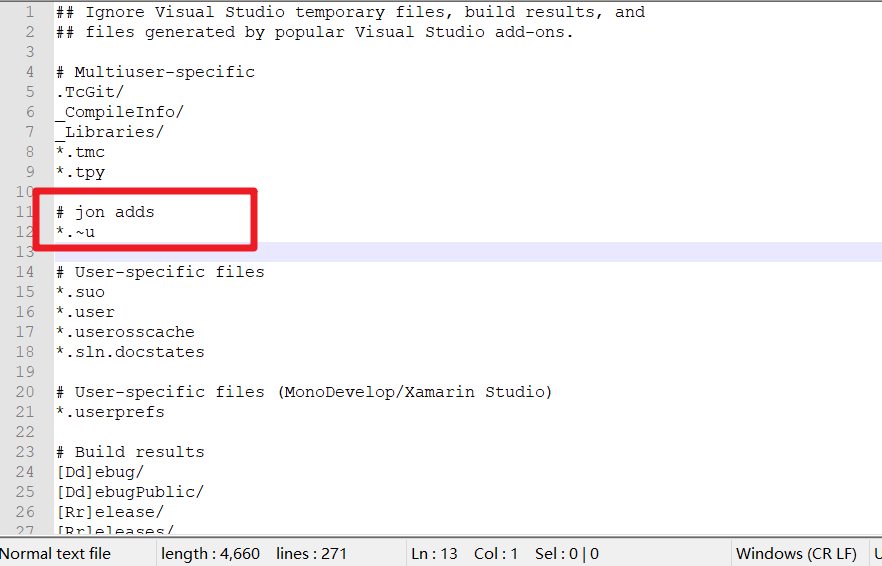
1. 手动修改“.gitignore”文件

还可以通过notepad++等工具打开“.gitignore”文件，把TwinCAT编译及备份自动生成的文件手动添加到“.gitignore”文件中，这样git会自动忽略这部分文件，从而不会对git的版本管理产生影响。

在3.2章d小节中，由于已经把TwinCAT Project1.project.~u文件通过git add . 命令添加到版本库追踪列表里了，我们第一步，通过git rm -f \*.~u命令将TwinCAT Project1.project.~u文件剔除出追踪。



然后在“.gitignore”文件中添加下图红框内容，就可以让git版本库中忽略所有的以~u结尾的文件。



## Git分支管理

当开发一个大型项目时，会有多人共同协作开发，Git分支管理提供了新建分支，合并分支，解决冲突等多种功能。以下通过简单的例子，介绍Git 分支管理的相关能容。

1. 创建及切换分支

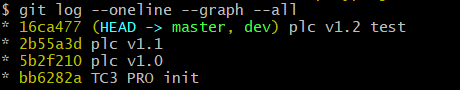
我们可以在git bash中通过git branch dev命令，创建一个新的分支。其中dev代表新分支的名称。

然后通过git branch -av命令，查看当前仓库的分支情况，如下：



我们可以看到当前仓库有2个分支，其中master是主分支，当创建一个仓库时，会自动产生master分支，dev即是刚创建的新分支。master分支前有一个 \* 号代表当前分支，在git bash中执行的命令都是针对当前分支有效。

然后通过git log –-oneline –-graph --all命令，查看当前仓库的log情况,我们可以看到当前master分支和dev分支都指向了16ca477这次commit，说明2个分支内容相同，没有变动。



通过git checkout dev命令，切换到dev分支，通过git branch -av命令可以看到\* 号换到了dev前面，代表分支千切换成功，如下图：

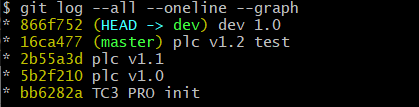


1. 在dev分支中创建新的commit

我们在仓库根目录下新建一个dev.txt文件，并添加文本’this is a dev file’,保存dev.txt文件。

然后通过git add .命令和git commit -m ‘dev 1.0’命令，创建一次提交。

然后通过git log –-oneline –-graph --all命令，查看当前仓库的log情况，如下



1. 切换回master，并添加新commit

通过git checkout master命令,切换回master分支。

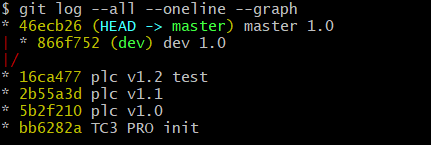


发现在仓库根目录下的dev.txt文件不见了。因为dev.txt文件目前还不属于master分支。

我们在仓库根目录下新建一个master.txt文件，并添加文本’this is a master file’,保存master.txt文件。

然后通过git add .命令和git commit -m ‘master 1.0’命令，创建一次提交。

然后通过git log –-oneline –-graph --all命令，查看当前仓库的log情况，如下

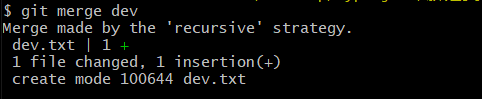


可以看到目前项目有2个不同的分支，不同的开发人员可以在不同的分支上单独进行开发，而互不影响。

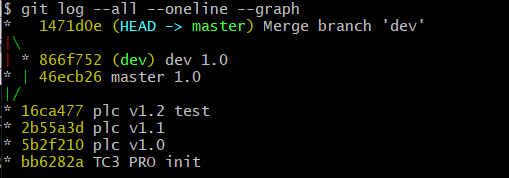
1. 合并分支

接下来我们把dev分支合并到master分支。

在master分支下，执行git merge dev命令，就可以把dev分支合并到master分支。执行git merge dev命令后，会弹出一个文本框，用来添加这次分支合并的版本描述信息，我们选择不修改，使用git提供的版本描述信息，直接关闭该文件。由于该例子不存在有冲突的文件（有冲突的文件指，2个分支都修改了某个文件，此时需要手动解决了冲突，才能合并），合并执行成功。如下图：



通过git log –-oneline –-graph --all命令，查看当前仓库的log情况，如下图：



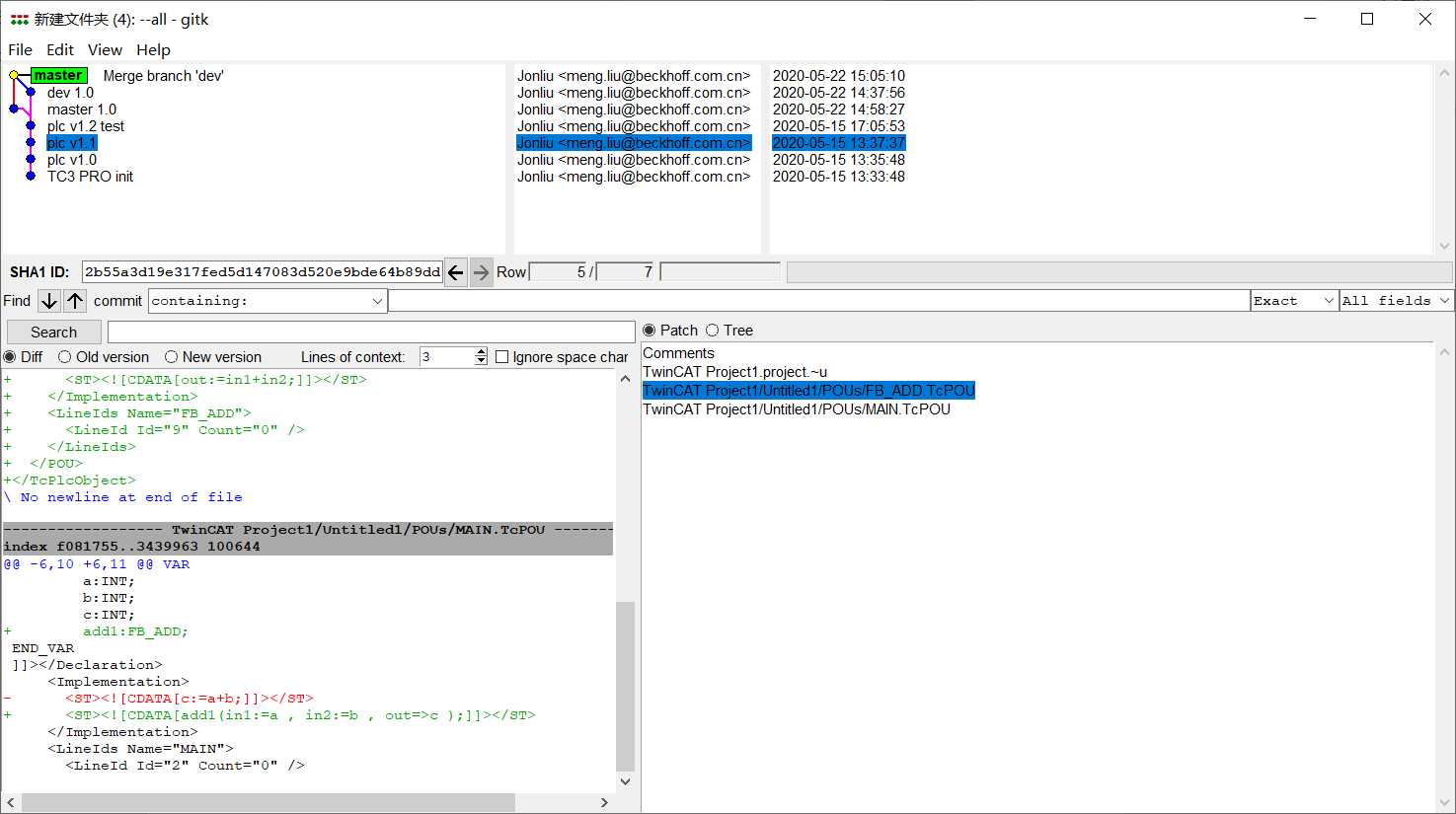
完成了代码合并，此时，我们可以通过git branch -d dev命令,删除掉dev分支了。



## 使用Git图形化操作工具gitk

Gitk是Git图形化操作工具，可以很清晰地查看搜索提交历史及 git 相关操作。之前我们都是通过git bash也就是命令行的方式来操作git。

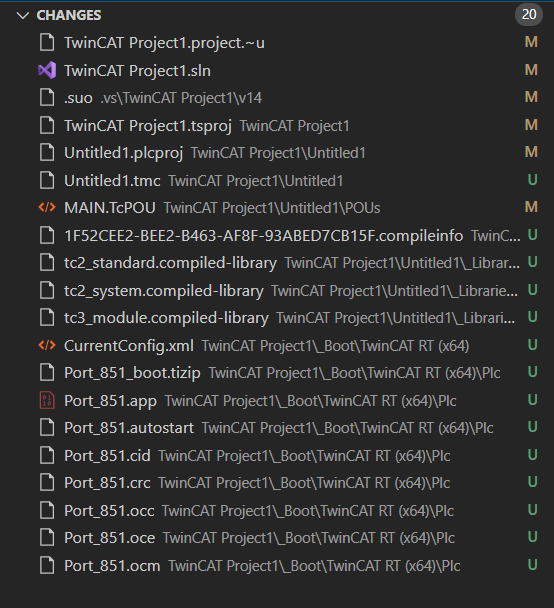
如何使用gitk？打开git代码仓库，右键git bash，在命令行中输入gitk --all即可打开gitk界面。在gitk中我们可以看到非常直观的看到版本库和分支的演变过程（下图左上）。也可以看到每次commit，都更改了哪些文件，以及具体变化的文本内容（下图左下）。



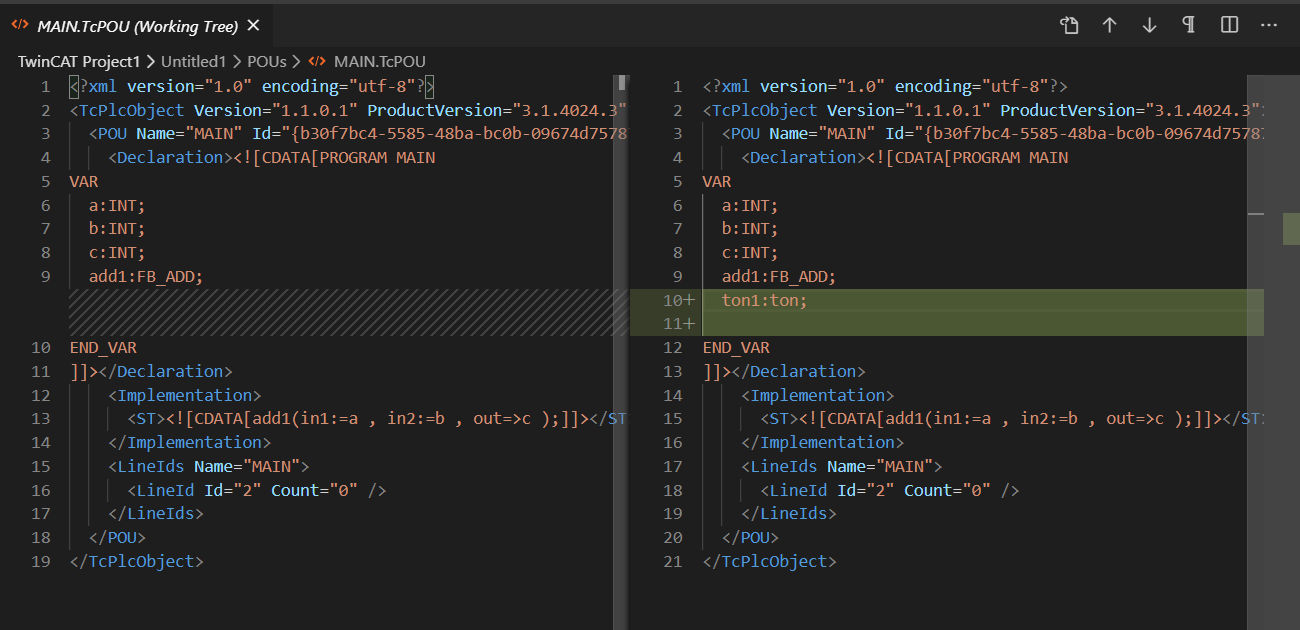
## 在VScode中使用git

在VScode开发工具中也集成了git的功能。相比较在命令行中使用git，VScode更能发挥git的文件比较，远程代码库管理等功能，非常的强大。

以文件比较为例，我们可以在VScode中看到每个文件的状态，U代表未追踪，M代表已修改。



双击MAIN.TcPOU,可以查看详细的文件对比，下图左为版本库的代码，右为当前工作区的代码。



在第二章节中讲述的所有命令也都可以在VScode中使用，有兴趣的可以自行研究。

# 连接远程仓库Github

## Github概述

Git是分布式版本控制系统，同一个Git仓库，可以分布到不同的机器上。怎么分布呢？最早，肯定只有一台机器有一个原始版本库，此后，别的机器可以“克隆”这个原始版本库，而且每台机器的版本库其实都是一样的，并没有主次之分。

Github可以充当一个服务器的角色，多人开发时，每个人都把本地的代码库提交到Github上，共同维护Github上的代码。

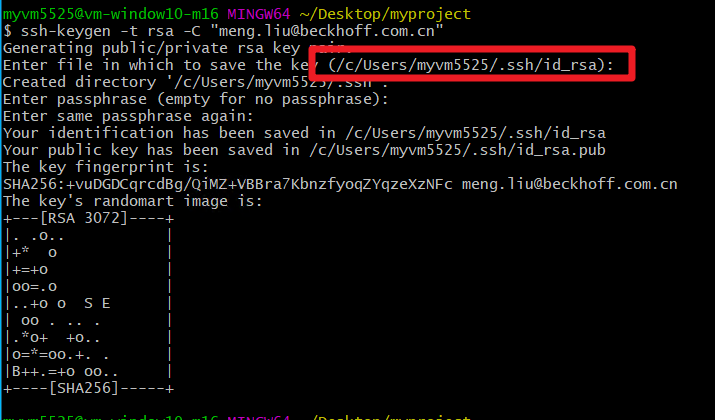
## 连接到远程仓库

后续内容需要有一个Github账号才可以继续测试。由于你的本地Git仓库和GitHub仓库之间的传输是通过SSH加密的，所以，需要做一些设置：

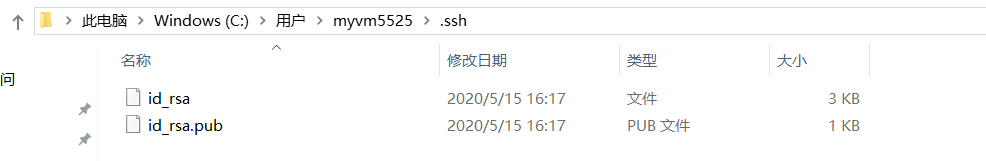
1. 创建SSH Key

打开Git Bash，输入ssh-keygen -t rsa -C "youremail@example.com"

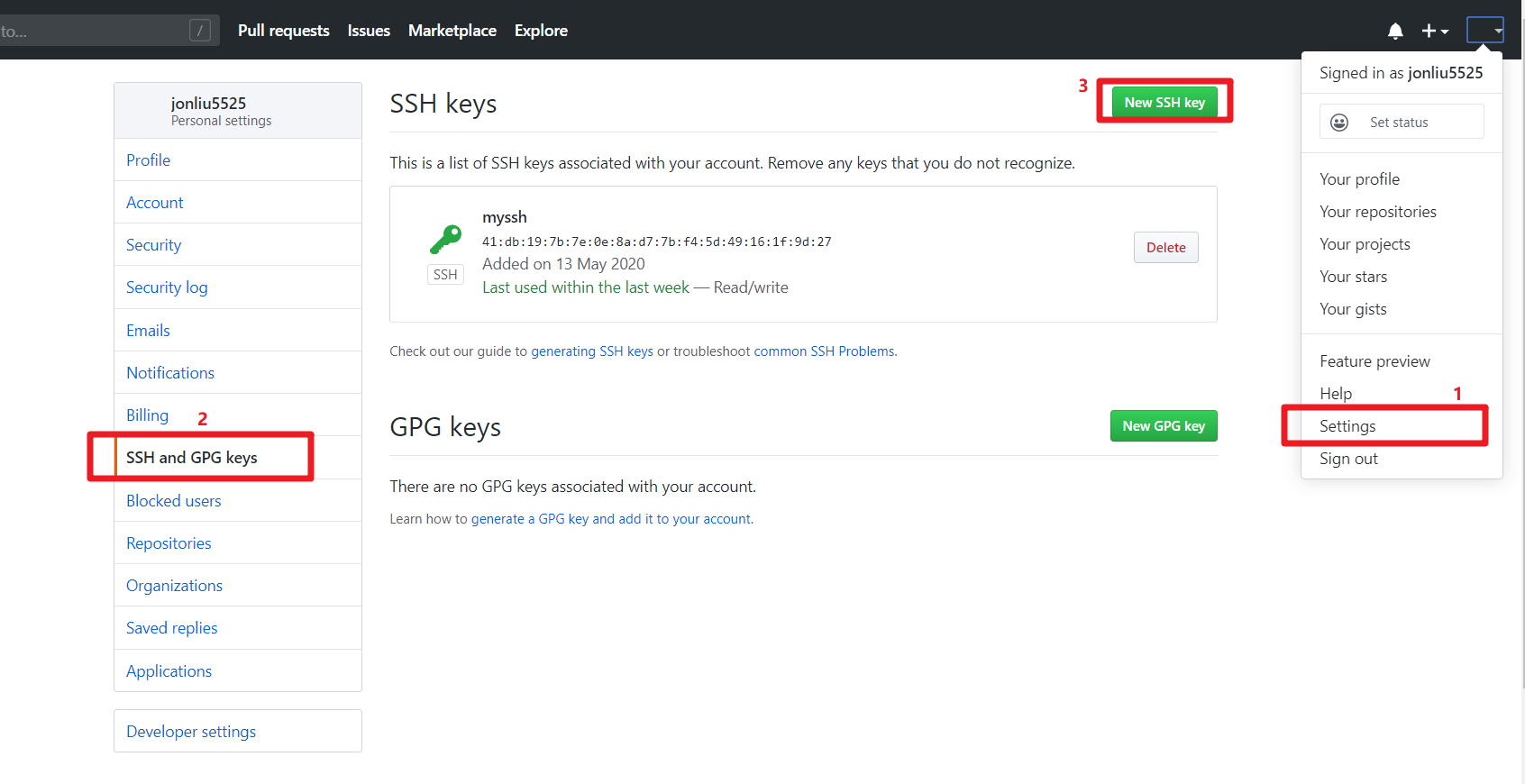
一直点击回车键即可。下图红框列出了ssh密匙的存储位置。



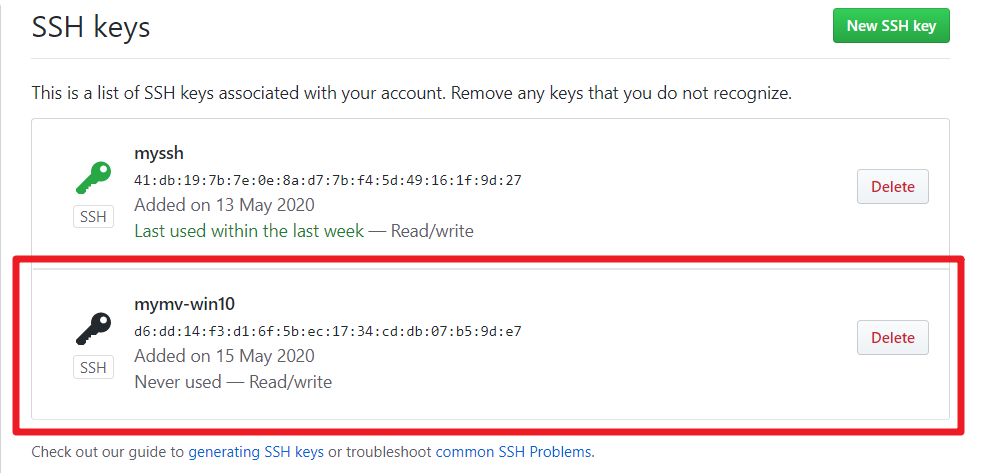
打开对应目录，找到相应文件，代表该操作执行成功。



1. 输入账号和密码，登录GitHub (github.com)，打开“settings”，“SSH and GPG Keys”页面，新建SSH key。



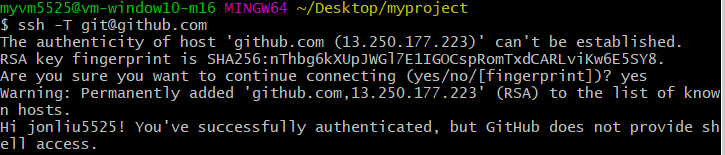
1. 填上任意Title，在Key文本框里粘贴id\_rsa.pub文件的内容，点击add ssh key。



1. SSH连接测试

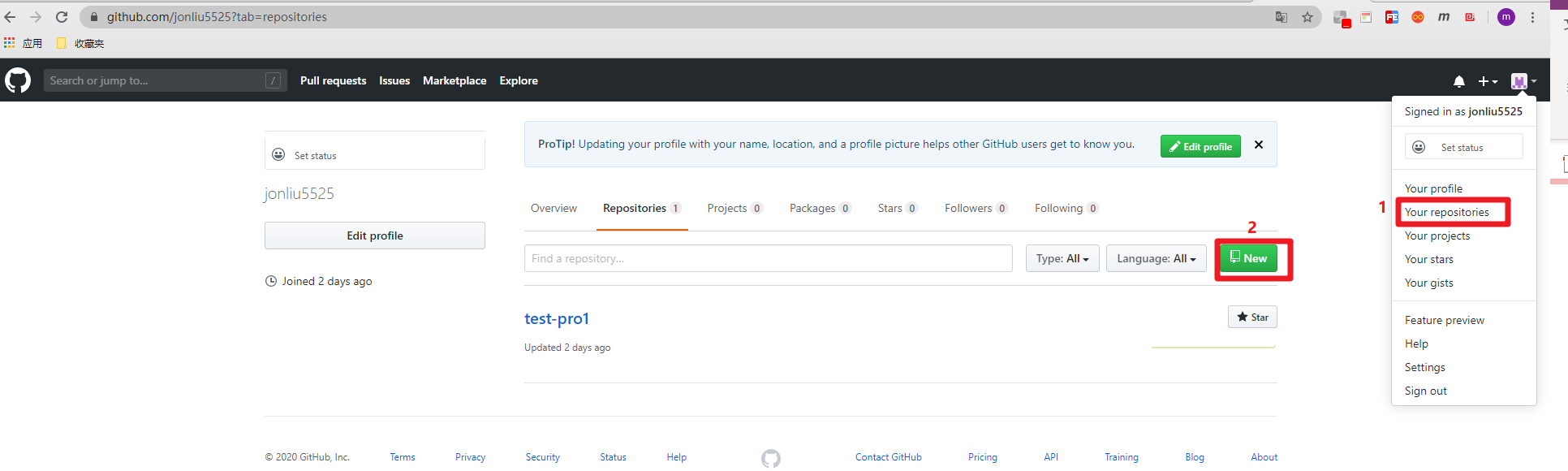
在git bash中输入ssh -T git@github.com ，然后在弹出的内容中，输入yes，回车。

以下内容代表连接成功。

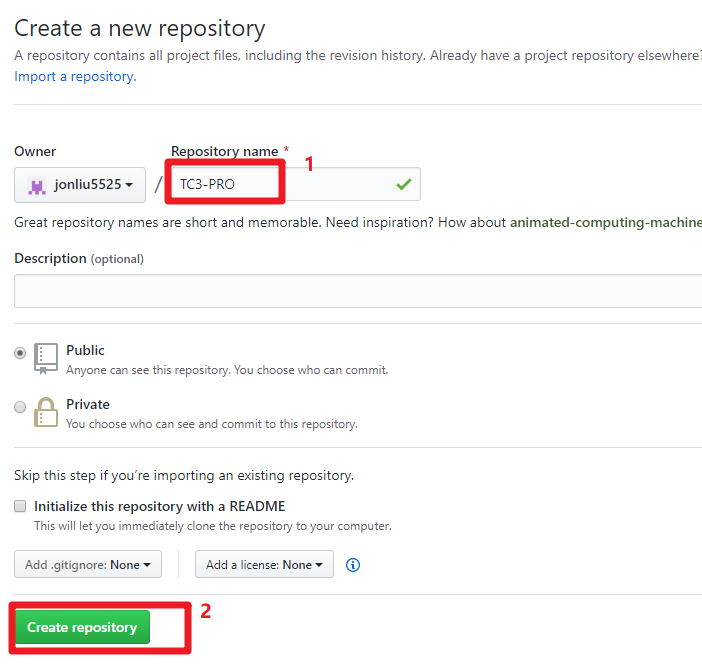


## 把本地工程上传到Github

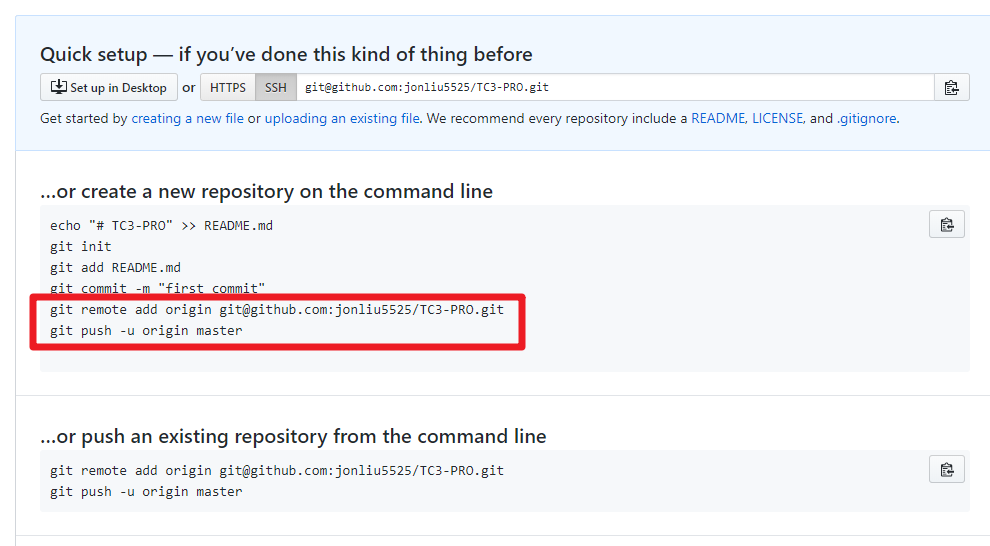
1. Github中新建一个仓库，如下图所示：



填入仓库名称，点击创建，如下图所示：



创建完成后的界面如下，我们把下图红框的2条命令复制下来：



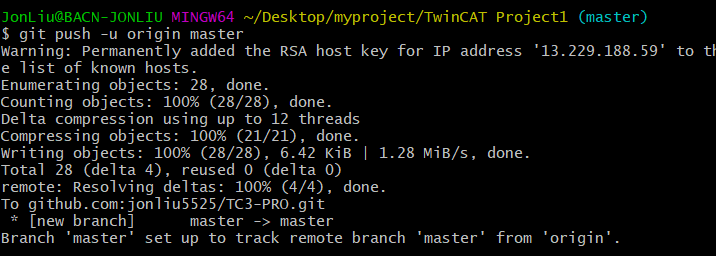
1. 把本地的工程代码上传到Github

回到桌面的..\myproject\TwinCAT Project1\目录，右键git bash。

将上图红框的git remote add origin [git@github.com:jonliu5525/TC3-PRO.git粘贴到 git](mailto:git@github.com:jonliu5525/TC3-PRO.git%20%20粘贴到%20git) bash中。该条命令进行本地仓库和远程仓库的关联。（此条命令内容以你github上复制的命令内容为准。）



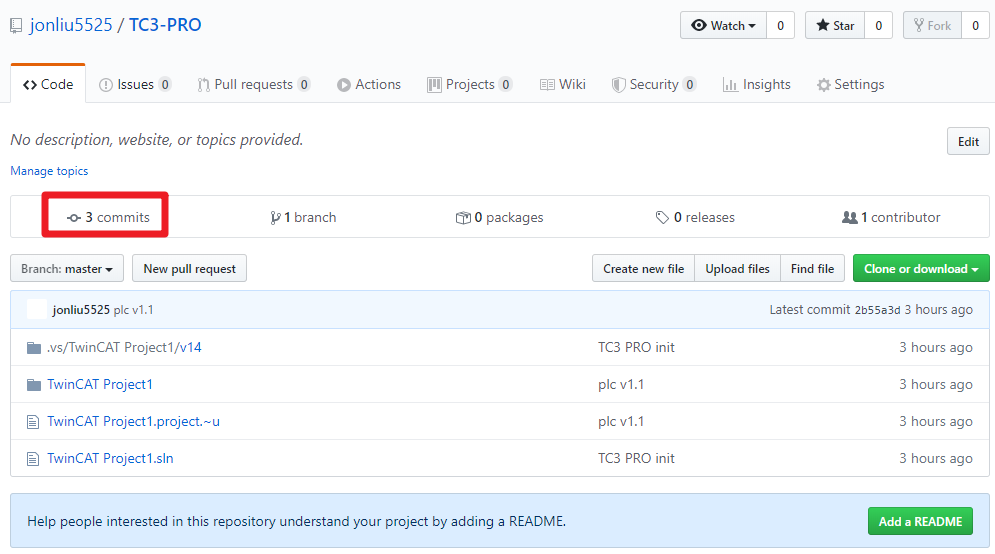
将上图红框的git push -u origin master粘贴到git bash中。该条命令进行本地仓库的文件上传。



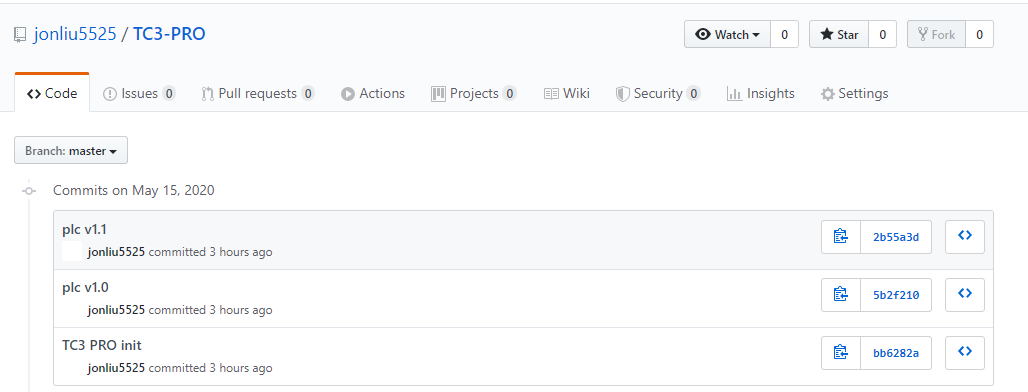
1. 在Github中查看工程代码

我们可以看到，在TC3-PRO仓库中已经把本地的文件同步了上去。

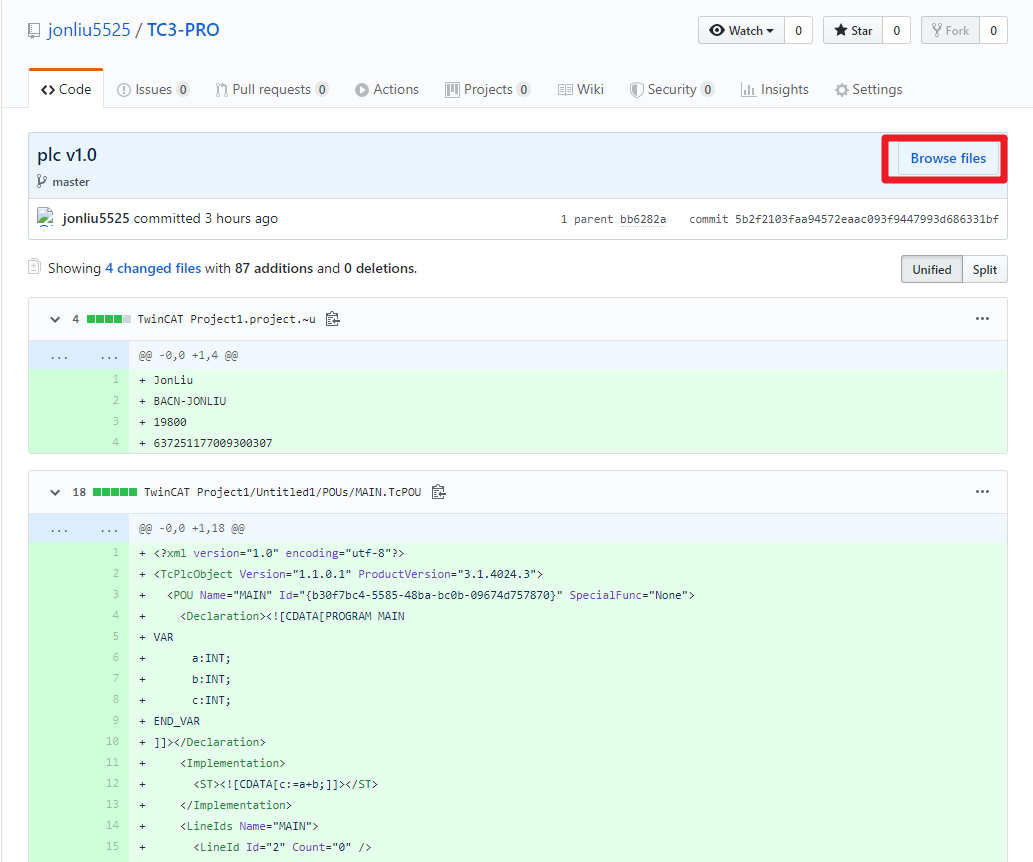
可以看到，该仓库共有3次提交，1个分支，还可以看到每个文件的commit信息。（本章节按照本地版本库commit plc v1.1进行的远程提交操作。）



点击上图中的3 commits，弹出如下内容：



点击任意版本，可以查看每次提交的修改内容。下图查看了plc v1.0的修改记录。



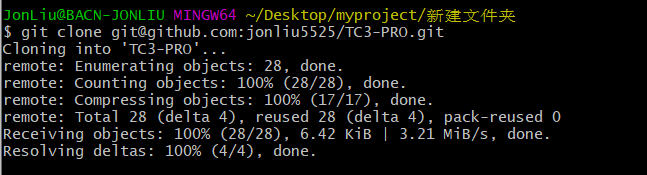
点击上图Browse files，可以查看任何一次commit时，当时的文件信息情况。

1. 拉取Github上仓库到本地

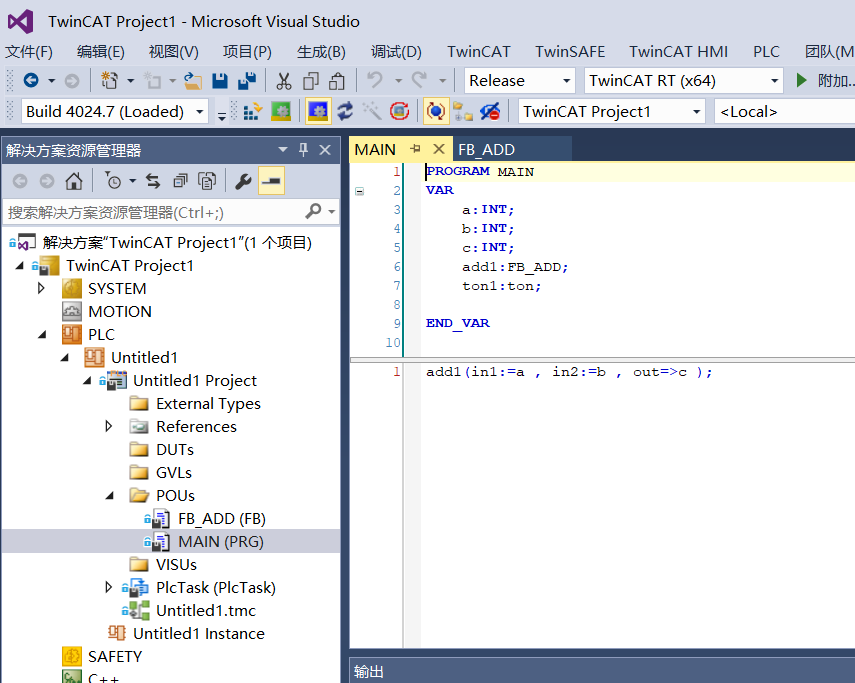
在桌面myproject/文件夹下右键git bash。

输入git clone [git@github.com:jonliu5525/TC3-PRO.git](mailto:git@github.com:jonliu5525/TC3-PRO.git)

即可将Github仓库的代码克隆到本地。（此条命令内容以你github的地址为准。）



打开TC3工程，与原工程内容相同。



通过以上操作我们完成本地和远程代码仓库的上传下载，可以实现多人多端的代码共享和协作编程工作。

# Git命令速查表

|  |  |  |
| --- | --- | --- |
| 功能 | 命令 | 说明 |
| 创建仓库 | git init |  |
| 把工作区文件添加到暂存区 | git add <文件名> | git add . 代表该仓库下的所有文件 |
| 把暂存区文件移除出追踪 | git rm <文件名> |  |
| 把暂存区内容提交到版本库 | git commit -m ‘版本描述’ |  |
| 查看当前版本库的状态 | git status |  |
| 查看提交的版本信息 | git log [--all] [--graph] [--oneline] | []内容可选 |
| 查看版本库操作记录 | git reflog |  |
| 版本回退 | git reset --hard <commit id> |  |
| 回退到上个版本 | git reset --hard HEAD^ |  |
| 暂存区恢复成和head一致 | git reset --hard HEAD |  |
| 丢弃工作区的修改 | git restore . | .代表所有文件 |
| 创建分支 | git branch <分支名> |  |
| 查看分支 | git branch -av |  |
| 切换分支 | git checkout <分支名> |  |
| 合并分支 | git merge <分支名> |  |
| 删除分支 | git branch -d <分支名> |  |
| 查看远程仓库信息 | git remote -v |  |
| 添加远程仓库 | git remote add <remote> <仓库地址> | 例子 git remote add origin git@github.com:jonliu5525/TC3-PRO.git |
| 拉取github仓库文件 | git pull --rebase <remote> <branch> | 例子 git pull –rebase origin master |
| 上传到github仓库 | git push -u <remote> <branch> | 例子 git push -u origin master |
| Github仓库克隆到本地 | git clone <仓库地址> |  |

**上海（ 中国区总部）**

中国上海市静安区汶水路 299 弄 9号（市北智汇园）

电话: 021-66312666 传真: 021-66315696 邮编：200072

**北京分公司**

北京市西城区新街口北大街 3 号新街高和大厦 407 室

电话: 010-82200036 传真: 010-82200039 邮编：100035

**广州分公司**

广州市天河区珠江新城珠江东路16号高德置地G2603室

电话: 020-38010300/1/2 传真: 020-38010303 邮编：510623

**成都分公司**

成都市锦江区东御街18号 百扬大厦2305 房

电话: 028-86202581 传真: 028-86202582 邮编：610016

|  |  |
| --- | --- |
| 请用微信扫描二维码  通过公众号与技术支持交流 |  |
| 倍福中文官网：  https://www.beckhoff.com.cn/ |
| 倍福虚拟学院：  https://tr.beckhoff.com.cn/ |
| 招贤纳士：job@beckhoff.com.cn  技术支持：support@beckhoff.com.cn  产品维修：service@beckhoff.com.cn  方案咨询：sales@beckhoff.com.cn |