目录

**[1. Git 添加所有文件](#_Toc12810_WPSOffice_Level1)** **[2](#_Toc12810_WPSOffice_Level1)**

**[2. git 提交后撤销提交到缓存区](#_Toc16628_WPSOffice_Level1)** **[2](#_Toc16628_WPSOffice_Level1)**

**[3. git 撤销缓存区到工作区](#_Toc16670_WPSOffice_Level1)** **[2](#_Toc16670_WPSOffice_Level1)**

**[4. git 撤销工作区的修改](#_Toc8845_WPSOffice_Level1)** **[2](#_Toc8845_WPSOffice_Level1)**

**[5. 删除文件命令](#_Toc14190_WPSOffice_Level1)** **[2](#_Toc14190_WPSOffice_Level1)**

**[6. git 跟踪删除文件](#_Toc28783_WPSOffice_Level1)** **[3](#_Toc28783_WPSOffice_Level1)**

[1. rm file后](#_Toc16628_WPSOffice_Level2) [3](#_Toc16628_WPSOffice_Level2)

[2. git rm file](#_Toc16670_WPSOffice_Level2) [3](#_Toc16670_WPSOffice_Level2)

[3. git commit –m ‘rm file’](#_Toc8845_WPSOffice_Level2) [3](#_Toc8845_WPSOffice_Level2)

**[7. 生成SHH Key](#_Toc13554_WPSOffice_Level1)** **[3](#_Toc13554_WPSOffice_Level1)**

**[8. 添加远程仓库](#_Toc21891_WPSOffice_Level1)** **[3](#_Toc21891_WPSOffice_Level1)**

**[9. 把本地提交内容推送到远程服务器](#_Toc13171_WPSOffice_Level1)** **[3](#_Toc13171_WPSOffice_Level1)**

**[10. 新建项目同步到（提交）远程失败](#_Toc31980_WPSOffice_Level1)** **[3](#_Toc31980_WPSOffice_Level1)**

[一、问题分析](#_Toc14190_WPSOffice_Level2) [4](#_Toc14190_WPSOffice_Level2)

[二、解决问题](#_Toc28783_WPSOffice_Level2) [4](#_Toc28783_WPSOffice_Level2)

**[11. 创建分支](#_Toc20506_WPSOffice_Level1)** **[4](#_Toc20506_WPSOffice_Level1)**

**[12. 删除分支](#_Toc31603_WPSOffice_Level1)** **[5](#_Toc31603_WPSOffice_Level1)**

**[13. 合并分支](#_Toc12203_WPSOffice_Level1)** **[5](#_Toc12203_WPSOffice_Level1)**

**[14. git用命令行实现从ssh切换到https](#_Toc16746_WPSOffice_Level1)** **[5](#_Toc16746_WPSOffice_Level1)**

[1、从ssh切换至https](#_Toc13554_WPSOffice_Level2) [5](#_Toc13554_WPSOffice_Level2)

[2、从https切换至ssh](#_Toc21891_WPSOffice_Level2) [5](#_Toc21891_WPSOffice_Level2)

[3、查看当前是ssh还是https](#_Toc13171_WPSOffice_Level2) [5](#_Toc13171_WPSOffice_Level2)

**[15. Git 上传忽略调跟踪的文件、文件夹](#_Toc19812_WPSOffice_Level1)** **[5](#_Toc19812_WPSOffice_Level1)**

**[16. 安装好SourceTree后，首次使用时，需要通过账户登录，但注册或登录界面可能根本无法打开，导致无法进入软件](#_Toc16515_WPSOffice_Level1)** **[9](#_Toc16515_WPSOffice_Level1)**

[解决办法](#_Toc31980_WPSOffice_Level2) [9](#_Toc31980_WPSOffice_Level2)

**[17. 配置sourcetree](#_Toc20980_WPSOffice_Level1)** **[11](#_Toc20980_WPSOffice_Level1)**

**[18. Git本地分支和远程分支如何关联](#_Toc3429_WPSOffice_Level1)** **[11](#_Toc3429_WPSOffice_Level1)**

## Git 添加所有文件

git add -A 添加所有变化

git add -u 添加被修改(modified)和被删除(deleted)文件，不包括新文件(new)

git add . 添加新文件(new)和被修改(modified)文件，不包括被删除(deleted)文件

## git 提交后撤销提交到缓存区

* 1. 如果保留修改的撤销->只删掉提交记录<- ： git reset HEAD^ 或者 git reset XXX(上一次哈希值)
  2. $ git reset --soft 5029f0cc08cf（上一次哈希值） //保修修改
  3. 如果把修改记录也干掉，用git reset --hard HEAD^

## git 撤销缓存区到工作区

可以把暂存区的修改撤销掉（unstage），重新放回工作区

file(要撤销修改的文件名如readme.txt)

git reset HEAD file

## git 撤销工作区的修改

file(要撤销修改的文件名)

git checkout – file

## 删除文件命令

rm file

## git 跟踪删除文件

* 1. rm file后
  2. git rm file
  3. git commit –m ‘rm file’

## 生成SHH Key

ssh-keygen -t rsa -C "897380685@qq.com"

## 添加远程仓库

git remote add origin git@github.com:michaelliao/learngit.git

## 把本地提交内容推送到远程服务器

git push -u origin master

由于远程库是空的，我们第一次推送master分支时，加上了-u参数，Git不但会把本地的master分支内容推送的远程新的master分支，还会把本地的master分支和远程的master分支关联起来，在以后的推送或者拉取时就可以简化命令。

## 新建项目同步到（提交）远程失败

wbf[@wangbaofeng](https://my.oschina.net/wangbaofeng" \t "https://my.oschina.net/wangbaofeng/blog/_blank) MINGW64 /e/gitProjects/myDemoPro (master)  
$ git push origin master  
To git@github.com:zuiwangzuo/myDemoPro.git  
 ! [rejected]        master -> master (fetch first)  
error: failed to push some refs to 'git@github.com:zuiwangzuo/myDemoPro.git'  
hint: Updates were rejected because the remote contains work that you do  
hint: not have locally. This is usually caused by another repository pushing  
hint: to the same ref. You may want to first integrate the remote changes  
hint: (e.g., 'git pull ...') before pushing again.  
hint: See the 'Note about fast-forwards' in 'git push --help' for details.

****一、问题分析****

先到github上看看远程repository下到底有啥东西，发现目标库是空的。

但是有一个README.md文件（本地代码中没有），而出现错误的原因是github中的README.md文件不在本地代码目录中。

也就是说我们需要先将远程代码库中的任何文件先pull到本地代码库中，才能push新的代码到github代码库中。

****二、解决问题****

1. 将远程代码库中的代码pull到本地代码库，并且执行合并merge：

$ git pull --rebase origin master  
warning: no common commits  
remote: Counting objects: 3, done.  
remote: Total 3 (delta 0), reused 3 (delta 0), pack-reused 0  
Unpacking objects: 100% (3/3), done.  
From github.com:zuiwangzuo/myDemoPro  
 \* branch            master     -> FETCH\_HEAD  
 \* [new branch]      master     -> origin/master  
First, rewinding head to replay your work on top of it...  
Applying: 提交初始代码。

2. 再次push代码：

$ git push origin master  
Counting objects: 20, done.  
Delta compression using up to 2 threads.  
Compressing objects: 100% (12/12), done.  
Writing objects: 100% (20/20), 2.51 KiB | 0 bytes/s, done.  
Total 20 (delta 0), reused 0 (delta 0)  
To git@github.com:zuiwangzuo/myDemoPro.git  
   8227f7c..aff8663  master -> master

我们看到本地代码已经成功的提交到远程代码中，至此问题解决。

## 创建分支

git checkout –b dev === git branch dev + git checkout dev

## 删除分支

git branch –d dev

## 合并分支

git merge dev

解决冲突

修改完冲突行后重新提交

## git用命令行实现从ssh切换到https

1、从ssh切换至https

git remote set-url origin(远程仓库名称) https://email/username/ProjectName.git

2、从https切换至ssh

git remote set-url origin git@email:username/ProjectName.git

3、查看当前是ssh还是https

git remote -v

## Git 上传忽略调跟踪的文件、文件夹

1、在需要创建 .gitignore 文件的文件夹, 右键选择Git Bash 进入命令行，进入项目所在目录。

2、输入 touch .gitignore ，生成“.gitignore”文件。

3、在”.gitignore” 文件里输入你要忽略的文件夹及其文件就可以了。（注意格式）

.DS\_Store

node\_modules/

dist/

npm-debug.log

1. 下面我们看看常用的规则：  
   1）/mtk/               过滤整个文件夹  
   2）\*.zip                过滤所有.zip文件  
   3）/mtk/do.c         过滤某个具体文件

node\_modules/ 表示过滤这个文件夹

\*.zip 过滤zip后缀文件

demo.html 过滤该文件

!src/ 不过滤该文件夹

!\*.js 不过滤java源文件

!index.html 不过滤该文件

1）配置语法：  
以斜杠“/”开头表示目录；  
以星号“\*”通配多个字符；  
以问号“?”通配单个字符  
以方括号“[]”包含单个字符的匹配列表；  
以叹号“!”表示不忽略(跟踪)匹配到的文件或目录；

此外，git 对于 .ignore 配置文件是按行从上到下进行规则匹配的，意味着如果前面的规则匹配的范围更大，则后面的规则将不会生效；

2）示例说明  
a）规则：fd1/\*  
说明：忽略目录 fd1 下的全部内容；注意，不管是根目录下的 /fd1/ 目录，还是某个子目录 /child/fd1/ 目录，都会被忽略；  
b）规则：/fd1/\*  
说明：忽略根目录下的 /fd1/ 目录的全部内容；  
c）规则：  
/\*  
!.gitignore  
!/fw/bin/  
!/fw/sf/  
说明：忽略全部内容，但是不忽略 .gitignore 文件、根目录下的 /fw/bin/ 和 /fw/sf/ 目录；

5、操作方法

一是常规的windows操作

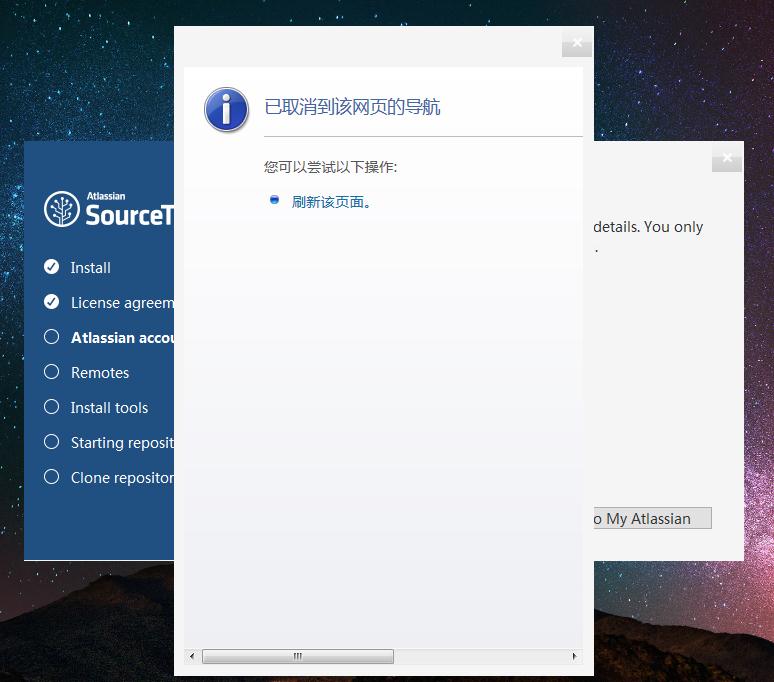
1. 根目录下创建gitignore.txt；
2. 编辑gitignore.txt，写下你的规则，例如加上node\_modules/；
3. 打开命令行窗口，切换到根目录（可以直接在文件夹上面的地址栏输入cmd回车）；
4. 执行命令ren gitignore.txt .gitignore。

二是用Git Bash

1. 根目录下右键选择“Git Bash Here”进入bash命令窗口；
2. 输入vim .gitignore命令，打开文件（没有文件会自动创建）；
3. 按i键切换到编辑状态，输入规则，例如node\_modules/，然后按Esc键退出编辑，输入:wq保存退出。

最后需要强调的一点是，如果你不慎在创建.gitignore文件之前就push了项目，那么即使你在.gitignore文件中写入新的过滤规则，这些规则也不会起作用，Git仍然会对所有文件进行版本管理。  
简单来说，出现这种问题的原因就是Git已经开始管理这些文件了，所以你无法再通过过滤规则过滤它们。因此一定要养成在项目开始就创建.gitignore文件的习惯，否则一旦push，处理起来会非常麻烦。

## 安装好SourceTree后，首次使用时，需要通过账户登录，但注册或登录界面可能根本无法打开，导致无法进入软件



## 解决办法

在目录C:\Users\{youruser}\AppData\Local\Atlassian\SourceTree 下创建文件accounts.json ，注意：{youruser}需要替换为登录系统用户名。

写入如下内容：

[

{

"$id": "1",

"$type": "SourceTree.Api.Host.Identity.Model.IdentityAccount, SourceTree.Api.Host.Identity",

"Authenticate": true,

"HostInstance": {

"$id": "2",

"$type": "SourceTree.Host.Atlassianaccount.AtlassianAccountInstance, SourceTree.Host.AtlassianAccount",

"Host": {

"$id": "3",

"$type": "SourceTree.Host.Atlassianaccount.AtlassianAccountHost, SourceTree.Host.AtlassianAccount",

"Id": "atlassian account"

},

"BaseUrl": "https://id.atlassian.com/"

},

"Credentials": {

"$id": "4",

"$type": "SourceTree.Model.BasicAuthCredentials, SourceTree.Api.Account",

"Username": "username@email.com"

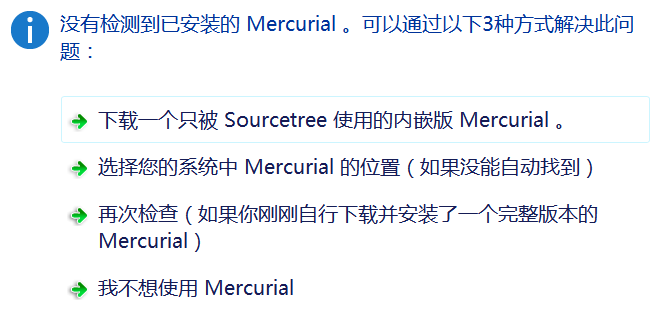
},

"IsDefault": false

}

]

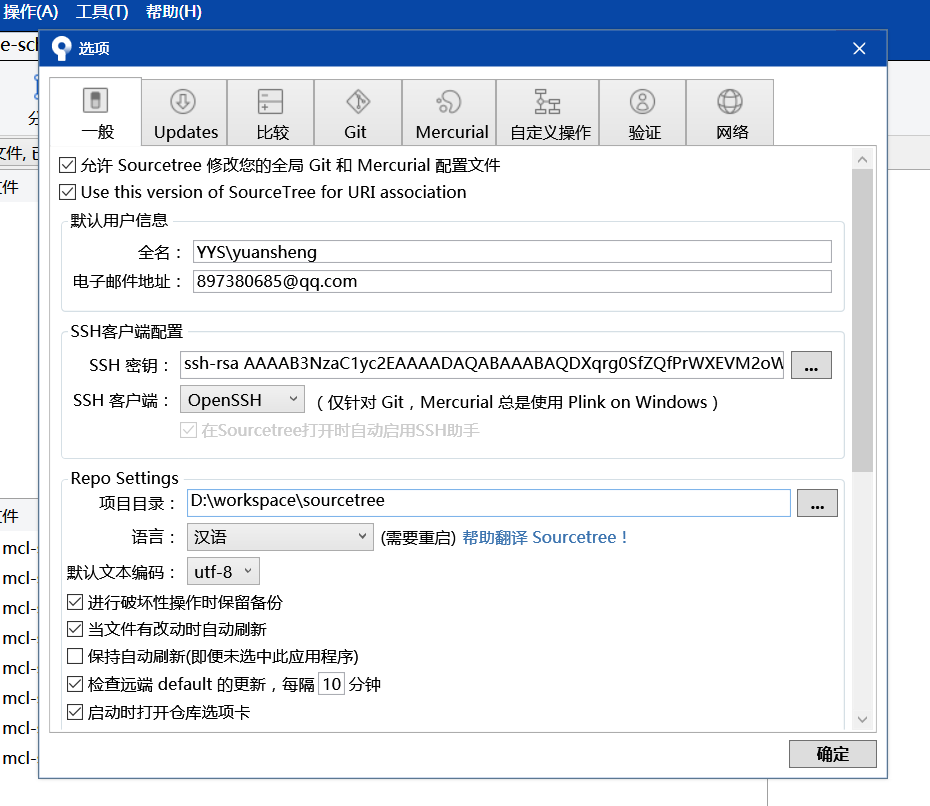
重新启动软件，顺利进入界面 ，如果有一下提示：



请到Mercurial官网下载安装：<https://www.mercurial-scm.org/>

## 配置sourcetree

参考网址：<https://www.cnblogs.com/wdlhao/p/9022511.html>



## Git本地分支和远程分支如何关联

****总结：****

****一般我们就用**git push --set-upstream origin branch\_name**来在远程创建一个与本地** branch\_name**同名的分支并跟踪；****

****利用**git checkout --track origin/branch\_name**来在本地创建一个与**branch\_name**同名分支 跟踪远程分支。****