

目录

前言	1.1
Git概述	1.2
基本操作	1.3
git设置	1.4
.gitignore	1.4.1
git的config	1.4.2
给git加代理	1.4.2.1
常见操作	1.5
新建仓库后如何操作	1.5.1
记住密码	1.5.2
迁移仓库且保留历史记录	1.5.3
PR(Pull Request)	1.5.4
常见问题	1.6
Updated upstream Stashed changes	1.6.1
fatal Authentication failed for	1.6.2
unable to access Empty reply from server	1.6.3
error failed to push some refs to	1.6.4
warning templates not found	1.6.5
changes would overwritten by merge	1.6.6
error RPC failed HTTP 504 curl 22	1.6.7
git应用	1.7
相关支持	1.7.1
git的IDE	1.7.2
VSCode	1.7.2.1
在线git仓库	1.7.3
基于git的系统	1.7.4
附录	1.8
相关教程	1.8.1
参考资料	1.8.2

最流行的版本管理系统：Git

- 最新版本： v1.0
- 更新时间： 20210423

简介

介绍目前最流行的版本控制管理系统git。先概述git，再介绍基本操作，包括代码的提交、同步、撤销等；详细介绍git的配置，包括config和.gitignore，尤其是config有本地和全局，以及相关的配置文件.git/config和.gitconfig，以及如何查看和修改配置。且对于常见的git的代理操作给出了详细的解释和操作；另外给出常见的操作，比如新建仓库后如何操作、记住密码、迁移仓库且保留历史记录、PR等；以及整理了一些常见问题，比如Updated upstream Stashed changes、fatal Authentication failed for、unable to access Empty reply from server、error failed to push some refs to、warning templates not found、changes would overwritten by merge、error RPC failed HTTP 504 curl 22 等等；整理出git相关应用，相关的支持、git的IDE、在线的git仓库系统、基于git的系统等。最后给出不错的git相关教程。

源码+浏览+下载

本书的各种源码、在线浏览地址、多种格式文件下载如下：

Gitbook源码

- [crifan/popular_version_control_git: 最流行的版本管理系统：Git](#)

如何使用此Gitbook源码去生成发布为电子书

详见：[crifan/gitbook_template: demo how to use crifan gitbook template and demo](#)

在线浏览

- [最流行的版本管理系统：Git book.crifan.com](#)
- [最流行的版本管理系统：Git crifan.github.io](#)

离线下载阅读

- [最流行的版本管理系统：Git PDF](#)
- [最流行的版本管理系统：Git ePUB](#)
- [最流行的版本管理系统：Git Mobi](#)

版权说明

给git加代理

此电子书教程的全部内容，如无特别说明，均为本人原创和整理。其中部分内容参考自网络，均已备注了出处。如有发现侵犯您版权，请通过邮箱联系我 admin 艾特 crifan.com，我会尽快删除。谢谢合作。

鸣谢

感谢我的老婆陈雪的包容理解和悉心照料，才使得我 crifan 有更多精力去专注技术专研和整理归纳出这些电子书和技术教程，特此鸣谢。

更多其他电子书

本人 crifan 还写了其他 100+ 本电子书教程，感兴趣可移步至：

[crifan/crifan_ebook_readme: Crifan的电子书的使用说明](#)

crifan.com，使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新：2021-04-23 20:26:16

Git概述

常见版本控制软件

版本管理 = 版本控制 的工具和软件，历史上有很多

之前听说或用过的有：

- 很早的： `perforce`
- 后来的： `svn`
- 最新的： `git`

大体区别如下：

版本控制系统SVN、Git、Perforce区别			
	SVN (Subversion)	Git	Perforce (P4)
适用场景	小公司	各种规模公司	大规模代码库： 如游戏公司、 动辄几十G的量产项目、 管理3D制作、音频文件
是否开源	开源	开源	商用
Server端	有一个Server端， 作为集中式版本库	有若干个Server端， 也有集中式版本库方便提交	—
适合开发模式	集中式开发	分布式开发	分布式开发
友好性	图形界面友好， 兼容各种系统类型	图形界面越趋友好	灵活的客户端视图
网络	内网	内外网	—
commit、查log	需要联网	可离线操作	—
branch	分支是一个简单的文件夹 分支的增删影响版本库	可以有无限个分支 分支可以合并 只要分支不合并和提交，不影响版本库	—
merge	较耗时	较省时	—
commit	先update，再commit，否则出错	commit不受同步先后的约束	—
版本库	支持部分检出	不支持部分检出， 每个克隆都是平等的	—
Server端故障	导致无法协同工作， 有丢失数据的风险	可用任何本地镜像(克隆)恢复版本库， 不影响协同工作	—

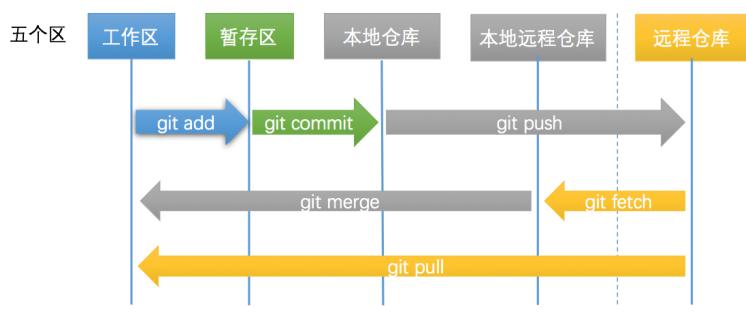
crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by
Gitbook最后更新： 2021-04-23 20:21:01

Git基本操作

代码提交和同步代码

最常见的基本操作：

- 概述



-

- 详解

- 最基本的

- 创建本地git仓库

```
git init
```

- 最常见的三步：新增并上传文件

- 添加文件

```
git add
```

- 提交

```
git commit
```

- 推送 = 上传到远端仓库

```
git push
```

- 更新文件

- 2步

- 下载新文件

```
git fetch
```

- 合并新文件

```
git merge
```

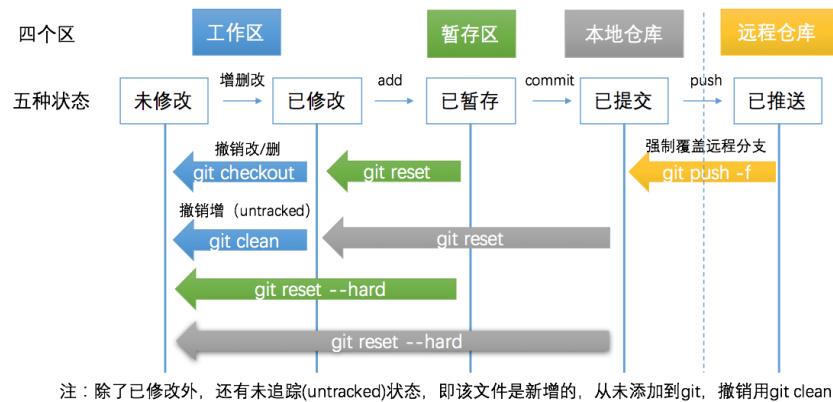
- 或：直接1步

- 下载并合并

```
git pull
```

代码撤销和撤销同步

相对高级一些的操作：



查看远程仓库url地址

```
git remote -v
```

其他操作

- 其他
 - 常见操作
 - 暂存本地更改 (往往在 git pull 之前)

```
git stash
```

- 恢复本地更改 (往往在 git pull 之后)

```
git stash pop
```

- 高级操作

- rebase

```
git rebase
```

- 分支

```
git branch  
git -b
```

要安装github中具体某个分支

格式：`git_address#branch_name`

举例：

```
npm install git://github.com/matthieu-prat/react-tappable.git#fix-is-mounted-de
```

给git加代理

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by
Gitbook最后更新: 2021-04-23 20:10:34

git设置

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by
Gitbook最后更新: 2021-04-23 20:22:31

.gitignore

- `.gitignore`
 - 是什么：普通的文本文件，是git中的一个配置文件
 - 作用：描述了 git 系统需要排除 ignore 哪些文件
 - 位置：
 - 最常见：git仓库根目录
 - 也可以：放在git仓库的任何子目录中

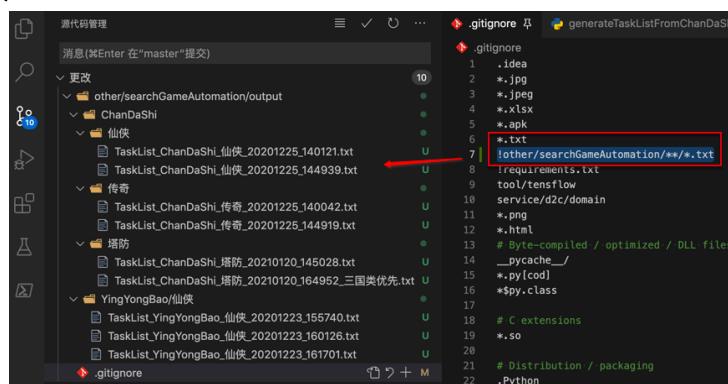
常见问题

排除掉其他所有，但只包含某个子目录中内容

- 举例1
 - 希望：忽略掉所有子目录中，所有txt文件
 - 但是不排除，即包含 `other/searchGameAutomation` 目录中的所有txt文件
 - `.gitignore` 的写法

```
*.txt
!other/searchGameAutomation/**/*.txt
```

- 效果



- 举例2
 - 希望：排除掉books下面所有的子文件夹
 - 但是只保留books/gitbook_demo
 - `.gitignore` 的写法

```
# exclude all in books
books/**/*
# include only single subfolder: gitbook_demo
!books/gitbook_demo
```

- 效果

给git加代理

```
.gitignore -- gitbook
4
5 *.zip
6
7 .DS_Store
8
9 generated/books/
10 generated/gitbook/
11
12 deploy_server_password.txt
13 # common/Config/deploy/deploy_server_password.txt
14
15 # exclude all in books
16 !books/**/*
17 # include only gitbook_demo
18 !books/gitbook_demo
19
20 !books/gitbook_demo

编辑 问题 输出 调试控制台 1: zsh
books/scientific_network_summary/
books/selenium_summary/
books/smarter_speaker_disassemble_summary/
books/super_search_regex/
books/xml_parse_write_spider/
books/xml_xpath_summary/
books/youdao_note_summary/
common/Makefile

no changes added to commit (use "git add" and/or "git commit -a")
crifan@licrifandeMacBook-Pro ~ ~/dev/dev_root/gitbook/GitbookTemplate/gitbook
行 20, 列 1 空格: 2 UTF-8 LF
```

- 相应的 git status 就没了那么多books了

```
终端 问题 输出 调试控制台 1: zsh
no changes added to commit (use "git add" and/or "git commit -a")
crifan@licrifandeMacBook-Pro ~ ~/dev/dev_root/gitbook/GitbookTemplate/gitbook_template
On branch master
Your branch is up to date with 'origin/master'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)
    modified:   .gitignore
    modified:   common/gitbook_makefile.mk

Untracked files:
  (use "git add <file>..." to include in what will be committed)

    common/Makefile

no changes added to commit (use "git add" and/or "git commit -a")
crifan@licrifandeMacBook-Pro ~ ~/dev/dev_root/gitbook/GitbookTemplate/gitbook_template
行 20, 列 1 空格: 2 UTF-8 LF Gitignore Go Live Rsync
```

排除项目根目录下data文件夹而保留某子文件夹中data文件夹

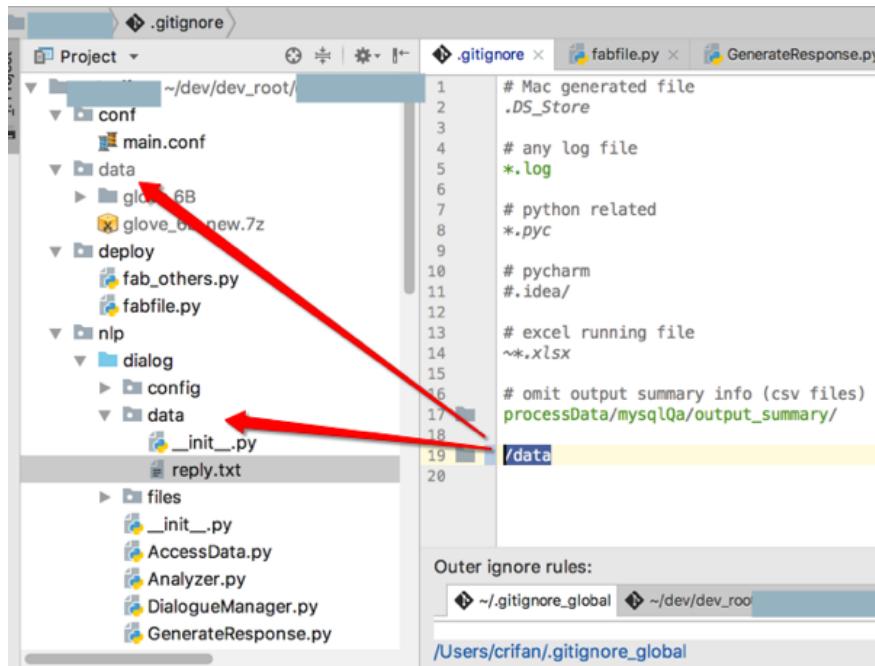
.gitignore 的配置：

```
/data
```

即可实现：

- 只排除掉项目根目录下的data文件夹
 - 但保留其他子文件夹中的data目录

效果：



不要误写成data/

如果误写成 data/ , 则会：

- 排除掉所有的 data文件夹
 - 除了会排除掉项目根目录下的data文件夹
 - 也会排除掉其他子文件夹中的data目录

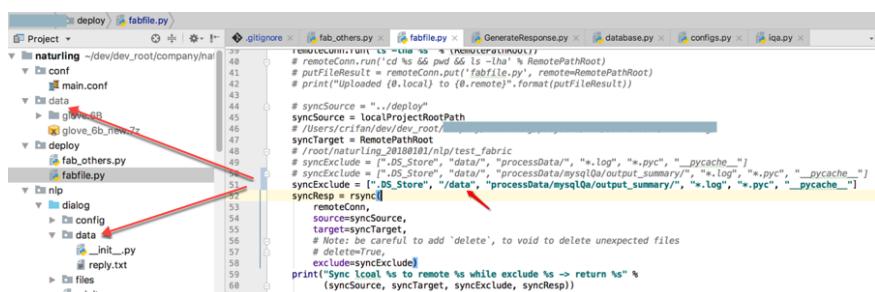


另外：

fabric中也是类似的逻辑

fabric 中利用 patchwork 的 rsync 去同步，在添加 exclude 参数，要排除掉的文件或文件夹时，语法也是类似的：

如果用 /data , 则也会导致子文件中的data目录被排除掉



给git加代理

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by
Gitbook最后更新: 2021-04-23 20:22:29

git的config

- git的配置

- 配置类型有2种

- 本地的

- 对应文件: `.git/config`

- 即当前 `.git` 目录下的 `config` 文件

- 命令行设置方式: 不加 `--global`

- 举例

- 查看本地配置

```
git config --list
```

- 取消本地代理

```
git config --unset http.proxy
```

- 会把 `.git/config` 中的 `http` 的 `proxy` 部分删除掉

- 全局的

- 对应文件: `~/.gitconfig`

- 命令行设置方式: 加 `--global`

- 举例

- 查看全局配置

```
git config --global --list
```

- 取消全局代理

```
git config --global --unset http.proxy
```

- 会把 `~/.gitconfig` 中的 `http` 的 `proxy` 部分删除掉

- 生效关系

- 优先级: 本地 > 全局

- 即: 本地的配置会覆盖全局的配置

- 配置的修改方式也有2种

- 直接修改配置文件

- 本地配置: 修改 `.git/config`

- 全局配置: 修改 `~/.gitconfig`

- 命令行方式设置参数

- 添加代理

- 添加本地代理

```
git config http.proxy socks5://127.0.0.1:1086
```

- 添加全局代理

```
git config --global http.proxy socks5://127.0.0.1:1086
```

- 取消代理

给git加代理

- 取消本地代理

```
git config --unset http.proxy
```

- 取消全局代理

```
git config --global --unset http.proxy
```

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by
Gitbook最后更新: 2021-04-23 20:22:02

给git加代理

有时候由于科学上网、下载速度慢等原因，需要去给git添加代理实现加速。

举例： [【已解决】github.io的git的push非常慢](#)

此处和Git的代理相关的操作有：

- 查看代理
- 设置代理=添加代理
- 取消代理

下面详细解释如何操作：

注：假如要设置的代理地址是：`socks5://127.0.0.1:1086`

- 查看（当前是否使用）代理

- 查看本地代理

- 方式

- 命令行

```
git config http.proxy
```

- 配置文件方式

```
cat .git/config
```

- 结果

- 可以看到：是否有 `http` 部分，`http` 中是否有 `proxy`，`proxy` 是否为空

- 查看全局代理

- 方式

- 命令行

```
git config --global http.proxy
```

- 配置文件方式

```
cat ~/.gitconfig
```

- 结果

- 可以看到：是否有 `http` 部分，`http` 中是否有 `proxy`，`proxy` 是否为空

- 设置（添加）代理

- 设置本地代理

- 方式

- 命令行

```
git config http.proxy socks5://127.0.0.1:1086
```

- 配置文件

给git加代理

```
vi .git/config
```

- 加上: http 的 proxy 的值是 socks5://127.0.0.1:1086

```
[http]
proxy = socks5://127.0.0.1:1086
```

- 设置全局代理

- 方式

- 命令行

```
git config --global http.proxy socks5://127.0.0.1:1086
```

- 配置文件

```
vi ~/.gitconfig
```

- 加上: http 的 proxy 的值是 socks5://127.0.0.1:1086

```
[http]
proxy = socks5://127.0.0.1:1086
```

- 取消代理

- 取消本地代理

- 方式

- 命令行

```
git config --unset http.proxy
```

- 配置文件

```
vi .git/config
```

- 方式1: 去掉 http 的 proxy

```
[http]
```

- 方式2: 设置 proxy 值是空

```
[http]
proxy =
```

- 取消全局代理

- 方式

- 命令行

```
git config --global --unset http.proxy
```

- 配置文件

```
vi ~/.gitconfig
```

- 方式1: 去掉 http 的 proxy

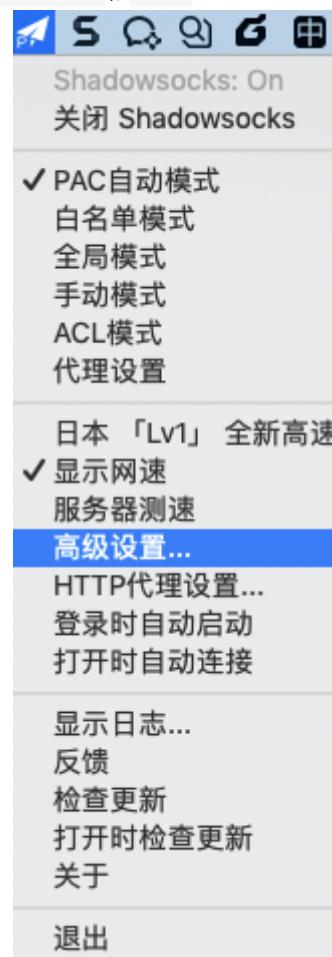


背景知识

关于自己电脑中可以使用的代理

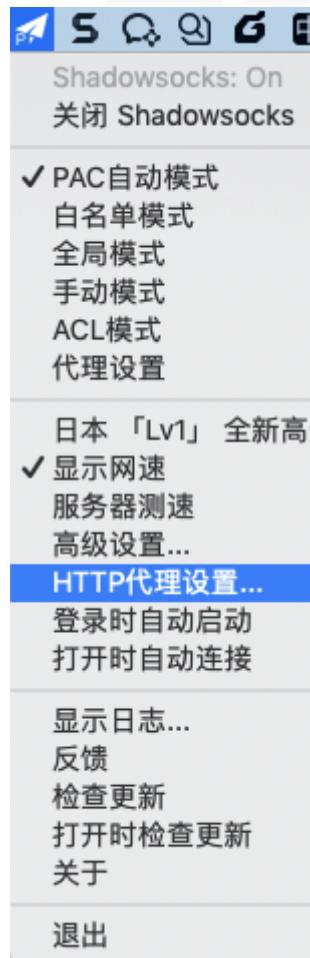
本地电脑中可以使用的代理，往往是像我一样，开启了科学上网的工具（ss / SSR / Trojan 等），所以有了：

- (默认开启的) **Socks5代理**
 - 举例： socks5://127.0.0.1:1086
 - 自己 Mac 中的 ShadowsocksX-NG 的 R 版 1.4.4-R8 (1)
 - 高级设置 -> 本地Sock5监听 地址和端口，分别是 127.0.0.1 和 1086





- 所以**Socks5**的代理地址就是:
 - socks5://127.0.0.1:1086
- (默认没开启, 要自己手动开启的) **http代理**
 - 举例: http://127.0.0.1:1087
 - 自己 Mac 中的 ShadowsocksX-NG 的 R 版 1.4.4-R8 (1)
 - HTTP代理设置 ->勾选: HTTP代理开启, 以及HTTP代理监听地址和端口, 分别是 127.0.0.1 和 1087





- 所以http的代理地址就是：
 - http://127.0.0.1:1087

注意事项

git的代理没有 https 的proxy，只有 http 的proxy

后经[git官网](#)证实：

- 结论：只有 http 的proxy，没有 https 的proxy
- 解释
 - (很多人) 以为
 - http.proxy 只针对 http://xxx 的http的网址
 - https.proxy 只针对 https://xxx 的https的网址
 - 比如常见的
 - https://github.com/xxx
 - https://gitee.com/xxx
 - 其实：此处 http.proxy 中的
 - http : 指的是HTTP协议，包括http和https的网址
 - proxy : 指的是代理，都加上代理
 - 所以：
 - 即使是（https://github.com、https://gitee.com 等）https的git的url地址，也是http的proxy，而不是https的proxy
 - 没有
 - 命令行中的写法

```
git config https.proxy
```

- 配置文件
 - 包括
 - 本地的： .git/config
 - 全局的： ~/.gitconfig
 - 中的写法

```
[https]
proxy = xxx
```

- 只有

给git加代理

- 命令行中的写法:

```
git config http.proxy
```

- 配置文件

- 包括

- 本地的: .git/config
 - 全局的: ~/.gitconfig

- 中的写法

```
[http]
proxy = xxx
```

特殊设置

单独针对某些git仓库=url 单独启用代理 或者 单独不用代理

举例：只给GitHub启用代理，其他不用代理

注：GitHub的地址是：<https://github.com>

- 命令行方式

- 本地代理

```
git config http.https://github.com.proxy socks5://127.0.0.1:1086
```

- 全局代理

```
git config --global http.https://github.com.proxy socks5://127.0.0.1:
```

- 配置文件方式

- 编辑配置文件

- 本地

```
vi .git/config
```

- 全局

```
vi ~/.gitconfig
```

- 文件内容

```
[http "https://github.com"]
proxy = socks5://127.0.0.1:1086
```

举例：其他全部启用代理（包括github），而gitee不用代理

- 命令行方式

- 本地

给git加代理

```
git config http.proxy socks5://127.0.0.1:1086  
git config https://gitee.com.proxy
```

- 全局

```
git config --global http.proxy socks5://127.0.0.1:1086  
git config --global https://gitee.com.proxy
```

- 配置文件方式

- 编辑配置文件

- 本地

```
vi .git/config
```

- 全局

```
vi ~/.gitconfig
```

- 文件内容

```
[http]  
proxy = socks5://127.0.0.1:1086  
[http "https://gitee.com/"]  
proxy =
```

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by
Gitbook最后更新: 2021-04-23 20:21:59

常见操作

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by
Gitbook最后更新: 2021-04-23 20:18:03

新建仓库后如何操作

新建git项目后，如何操作

简易的命令行入门教程：

git全局设置：

```
git config --global user.name "CrifanLi"  
git config --global user.email "crifan.li@xxx.com"
```

创建git仓库：

```
mkdir see_empty_project_git  
cd see_empty_project_git  
git init  
touch README.md  
git add README.md  
git commit -m "first commit"  
git remote add origin https://gitee.com/xxx_crifan/see_empty_project_git.git  
git push -u origin master
```

已有项目 = 把当前代码加到已有项目

```
cd existing_git_repo  
git remote add origin https://gitee.com/xxx_crifan/see_empty_project_git.git  
git push -u origin master
```

举例： EvernoteToWordpress

[crifan/EvernoteToWordpress](#)

快速设置—如果你知道该怎么操作，直接使用下面的地址

HTTPS SSH <https://gitee.com/crifan/EvernoteToWordp>

我们强烈建议所有的git仓库都有一个`README`, `LICENSE`, `.gitignore`文件

Git入门？查看 [帮助](#), [Visual Studio](#) / [TortoiseGit](#) / [Eclipse](#) / [Xcode](#) 下如何连接本站, 如何导入仓库

简易的命令行入门教程:

Git 全局设置:

```
git config --global user.name "crifan"  
git config --global user.email "admin@crifan.com"
```

创建 git 仓库:

```
mkdir EvernoteToWordpress  
cd EvernoteToWordpress  
git init  
touch README.md  
git add README.md  
git commit -m "first commit"  
git remote add origin https://gitee.com/crifan/EvernoteToWordpress.git  
git push -u origin master
```

已有仓库?

```
cd existing_git_repo  
git remote add origin https://gitee.com/crifan/EvernoteToWordpress.git  
git push -u origin master
```

```
简易的命令行入门教程：  
Git 全局设置：  
git config --global user.name "crifan"  
git config --global user.email "admin@crifan.com"  
  
创建 git 仓库：  
mkdir EvernoteToWordpress  
cd EvernoteToWordpress  
git init  
touch README.md  
git add README.md  
git commit -m "first commit"  
git remote add origin https://gitee.com/crifan/EvernoteToWordpress.git  
git push -u origin master  
  
已有仓库？  
cd existing_git_repo  
git remote add origin https://gitee.com/crifan/EvernoteToWordpress.git  
git push -u origin master
```

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by
Gitbook最后更新： 2021-04-23 20:24:02

记住密码

```
git config --global credential.helper store
```

之后，正常git操作，比如：

```
git pull
```

再输入账号和密码

-> git就会记住了

-> 下次再git操作，就不用再输入账号和密码了

crifan.com，使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by
Gitbook最后更新：2021-04-23 20:18:19

迁移仓库且保留历史记录

之前遇到过个需求：整体迁移git仓库，且保留所有历史commit提交记录

步骤是：

```
git clone --mirror old-repo-url new-repo  
  
cd new-repo  
git remote remove origin  
  
git remote add origin new-repo-url  
  
git push --all  
git push --tags
```

说明：

此处的：

```
git clone --mirror <url to ORI repo> temp-dir
```

等价于：

```
git clone <url to ORI repo> temp-dir  
  
git branch -a  
  
git checkout branch-name  
  
git fetch --tags  
  
git tag  
git branch -a
```

后记：确认和验证新仓库代码是正常的

```
cd ..  
rm -rf new-repo  
git clone new-repo-url new-repo
```

其中：把new-repo-url和 new-repo 换成你自己的仓库

举例：迁移appcrawler

此处自己的操作：

以镜像方式下载复制代码

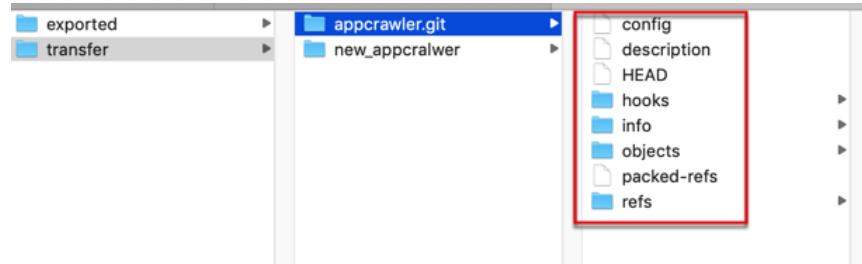
```
git clone --mirror http://xxx.xxx.com:yyy/data/data_limao/appcrawler.git appcr
```

下载后是git相关文件，而不是源码

给git加代理

此处下载后，本地文件夹中看到的内容，不是源码，而是git的一些文件：

```
cd appcrawler.git
limao@xxx ~/dev/xxx/gitlab/transfer/appcrawler.git master ll
total 32
-rw-r--r-- 1 limao CORP\Domain Users 23B 7 15 15:23 HEAD
-rw-r--r-- 1 limao CORP\Domain Users 238B 7 15 15:23 config
-rw-r--r-- 1 limao CORP\Domain Users 73B 7 15 15:23 description
drwxr-xr-x 13 limao CORP\Domain Users 416B 7 15 15:23 hooks
drwxr-xr-x 3 limao CORP\Domain Users 96B 7 15 15:23 info
drwxr-xr-x 4 limao CORP\Domain Users 128B 7 15 15:23 objects
-rw-r--r-- 1 limao CORP\Domain Users 105B 7 15 15:23 packed-refs
drwxr-xr-x 4 limao CORP\Domain Users 128B 7 15 15:23 refs
```



-> 不要和我之前一样误以为是操作失败了。这是正常的，期望的结果，不是出错了。

删除本地的远端的分支

```
cd appcrawler
git remote remove origin
```

其中会有提示，意思好像是需要你主动删除原有分支？总之可以忽略不管。

注：

```
git remote remove origin
```

的另一种写法：

```
git remote rm origin
```

添加远端地址为新仓库

```
git remote add origin http://xxx.corp.com:xxx/data_limao/appcrawler.git
```

提交上传所有代码和标签

- 上传所有代码：

```
git push --all
```

◦ 或

▪ 先

给git加代理

```
git push origin --all
```

- 和所有标签:

```
git push --tags
```

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by
Gitbook最后更新: 2021-04-23 20:24:21

PR(Pull Request)

参考：

- gitee = 码云 的git教程
 - 4.1、Fork + Pull 模式

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by
Gitbook最后更新：2021-04-23 20:18:00

常见问题

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by
Gitbook最后更新: 2021-04-23 20:16:42

Updated upstream Stashed changes

在

- 本地：已有文件改动
- 远程git仓库：也有文件改动
- 且本地改动和远程git仓库中的改动，是同一个文件，同一个（附近的）位置

的情况下，去

```
git pull
```

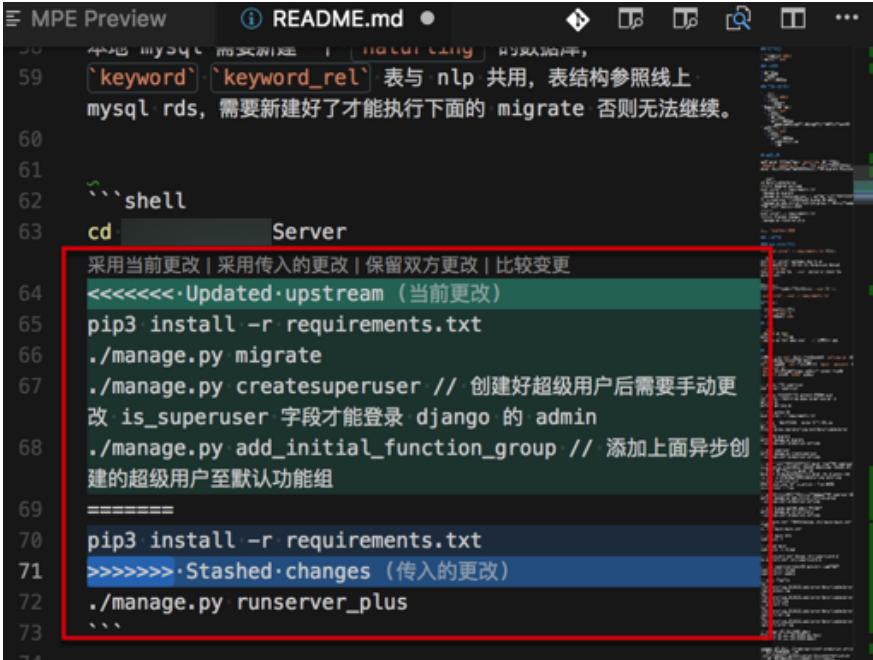
等操作时，往往会遇到：

```
<<<<<< Updated upstream
xxx
=====
yyy
>>>>>> Stashed changes
```

表示：出现冲突了

需要：自己根据实际情况去合并内容，取舍某个版本，或者合并2部分的内容，为最新的内容等等。

举例说明：



The screenshot shows a code editor window for a file named README.md. A conflict is highlighted with a red box. The conflict text is:

```
采用当前更改 | 采用传入的更改 | 保留双方更改 | 比较变更
<<<<<< Updated · upstream (当前更改)
64 pip3 install -r requirements.txt
65 ./manage.py migrate
66 ./manage.py createsuperuser // 创建好超级用户后需要手动更
改 is_superuser 字段才能登录 django 的 admin
67 ./manage.py add_initial_function_group // 添加上面异步创
建的超级用户至默认功能组
68 =====
69
70 pip3 install -r requirements.txt
71 >>>>>> Stashed · changes (传入的更改)
72 ./manage.py runserver_plus
73 ````
```

文件内容会变成：

给git加代理

```
cd XxxCmsServer
<<<<< Updated upstream
pip3 install -r requirements.txt
./manage.py migrate
./manage.py createsuperuser // 创建好超级用户后需要手动更改 is_superuser 字段才能登录
./manage.py add_initial_function_group // 添加上面异步创建的超级用户至默认功能组
=====
pip3 install -r requirements.txt
>>>>> Stashed changes
./manage.py runserver_plus
```

此处：由于本地内容更新，且已包含服务器=远端git仓库中的内容，则可以：

采用本地内容

则具体操作是：点击 采用当前更改，即可。

合并后的內容就是：

```
pip3 install -r requirements.txt
./manage.py migrate
./manage.py createsuperuser // 创建好超级用户后需要手动更改 is_superuser 字段才能登录
./manage.py add_initial_function_group // 添加上面异步创建的超级用户至默认功能组
```

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by
Gitbook最后更新：2021-04-23 20:16:38

fatal Authentication failed for

现象： git clone 或 git push 报错：

```
remote: You do not have permission to pull from the repository via HTTPS
fatal: Authentication failed for 'https://gitee.com/crifan/xxx.git/'
```

原因：（密码已更新导致了）之前本地保存的账号和密码不对（失效了）

解决办法：更换（更新）对应账号和密码

操作步骤：

```
git config --global user.name your_account_name
git config --global user.email your_email_address
```

举例

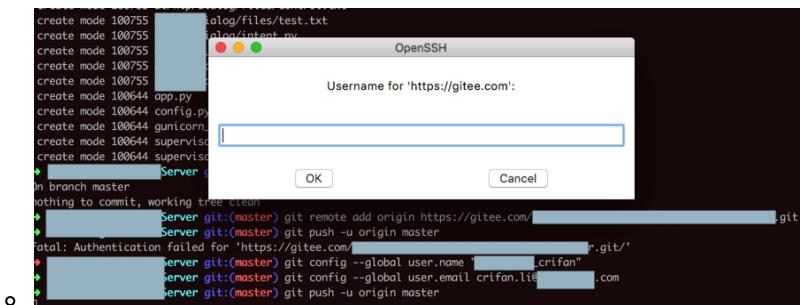
```
git config --global user.name crifan
git config --global user.email admin@crifan.com
```

然后再去

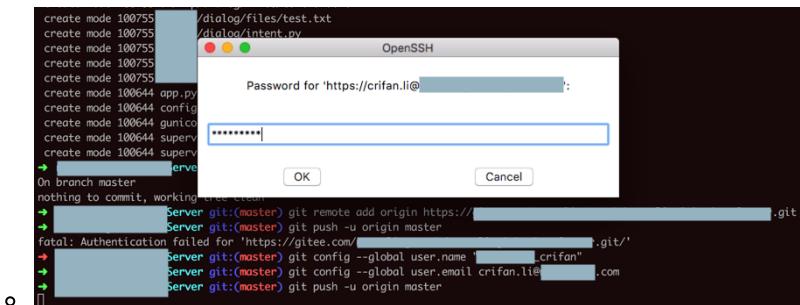
```
git clone
git push
```

等操作，即可弹框分别让你输入用户名和密码：

- 用户名



- 密码

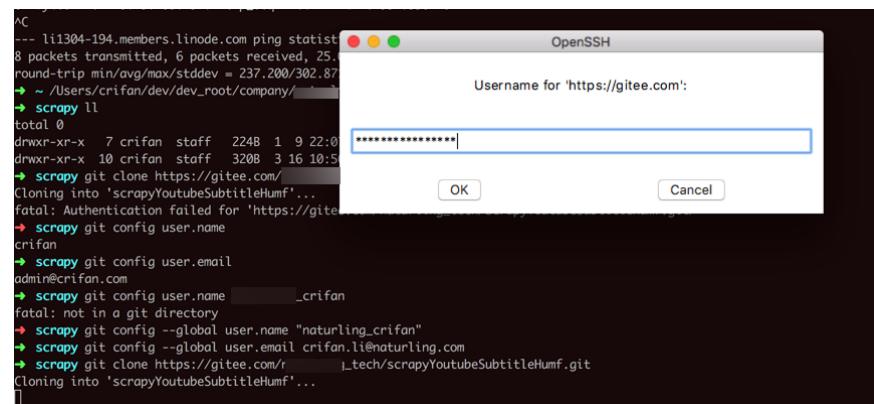


根据提示去输入，即可。

Mac的iTerm中可能看不到用户名和密码输入框

在Mac中iTerm的终端时，此处是看不到弹框的，需要

手动用快捷键 **Command + `** 去切换，才能看到（OpenSSH的）弹框窗口



否则，看不到弹框时，就会让人很懵，不知道怎么回事。

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by
Gitbook最后更新：2021-04-23 20:11:46

unable to access Empty reply from server

问题

在 git pull 或 git push , 报错:

```
fatal: unable to access Empty reply from server
```

原因

可能的原因有多种，比如网络断了等等。

此处遇到一种情况，其原因是：

当前git仓库是只允许内网访问，此处开启了代理，导致无法访问而报错

解决办法：去掉代理

操作步骤

- 清除git的本地代理

```
git config --unset http.proxy
```

- 清除git的全局代理

```
git config --global --unset http.proxy
```

查看git代理

- 查看git当前（本地）代理

```
git config http.proxy
```

- 查看git全局代理

```
git config --global http.proxy
```

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by
Gitbook最后更新: 2021-04-23 20:11:17

error failed to push some refs to

背景

在gitee上已经新建了git的repo仓库，且创建了一个README.md

而本地也新建了个git仓库，且加了内容：

```
git init  
git add  
git commit
```

然后去提交：

```
git push --set-upstream origin master
```

结果报错：

```
→ youtubeSubtitle git:(master) ✘ git push --set-upstream origin master  
To https://gitee.com/xxx/xxx.git  
! [rejected]          master -> master (non-fast-forward)  
error: failed to push some refs to 'https://gitee.com/xxx/xxx.git'  
hint: Updates were rejected because the tip of your current branch is behind  
hint: its remote counterpart. Integrate the remote changes (e.g.  
hint: 'git pull ...') before pushing again.  
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
```

原因：本地的内容，和远端仓库内容冲突了。

解决办法：如果像我此处一样，远端在线git仓库中内容是空的，被覆盖也无所谓。则可以采用：强制上传本地内容到在线仓库

操作步骤：

```
git push -u origin master -f
```

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by
Gitbook最后更新：2021-04-23 20:11:36

warning templates not found

现象： Mac 中用 git 的工具 SourceTree 去clone下载代码， 报错：

```
git -c filter.lfs.smudge= -c filter.lfs.required=false -c diff.mnemonicprefix=
Cloning into '/Users/minglong/saicgroup_srt-ios'...
warning: templates not found /usr/local/git/share/git-core/templates
error: RPC failed; HTTP 504 curl 22 The requested URL returned error: 504 Gate
fatal: The remote end hung up unexpectedly
Completed with errors, see above
```

原因：相关目录不存在

解决办法：创建对应目录，且加上权限

操作步骤：

```
mkdir -p /usr/local/git/share/git-core/templates
sudo chmod -R 755 /usr/local/git/share/git-core/templates
```

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by
Gitbook最后更新： 2021-04-23 20:16:53

changes would overwritten by merge

现象：`git pull` 出错：

```
xxxMacBook-Pro:xxx minglong$ git pull
Username for 'http://git.oschina.net': xxx@qq.com
Password for 'http://xxx@qq.com@git.oschina.net':
remote: Counting objects: 25, done.
remote: Compressing objects: 100% (25/25), done.
remote: Total 25 (delta 15), reused 0 (delta 0)
Unpacking objects: 100% (25/25), done.
From http://git.oschina.net/xxx/xxx
  dec7330..c99180f master      -> origin/master
    f0edd68..c99180f remote_push -> origin/remote_push
* [new tag]           remote_push -> remote_push
Updating dec7330..c99180f
error: Your local changes to the following files would be overwritten by merge
  JianDao/JianDao/Constants.swift
Please, commit your changes or stash them before you can merge.
Aborting
```

原因：本地有更新，但是没提交。而pull下载并合并，会把本地改动覆盖掉。

解决办法：把本地的暂存起来，再去pull更新并合并，再把暂存的恢复出来。

操作步骤：

```
git stash
git pull
git stash pop
```

补充：

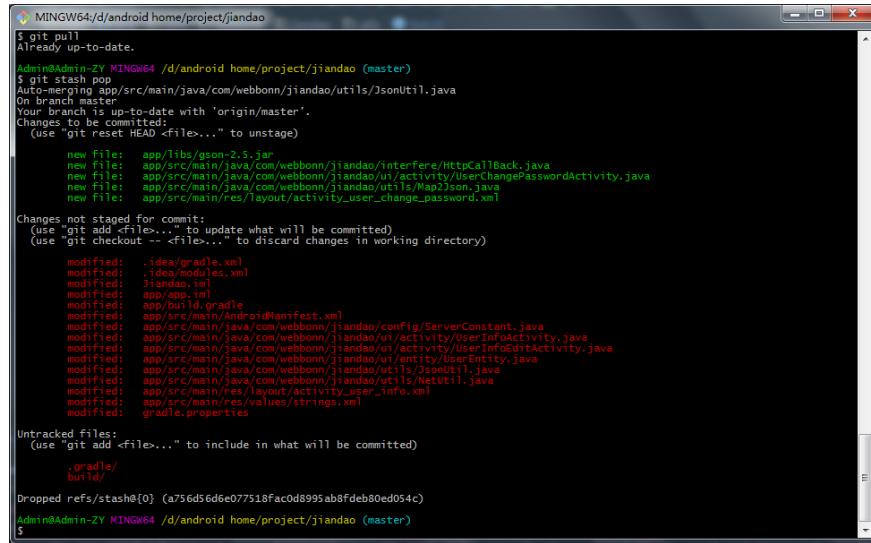
另外也遇到一个类似情况，但是处理方式是：

扔掉另外2个冲突的，但是无用的文件：

```
git checkout - file_you_want_throw_away
```

再去`git stash pop`即可：

给git加代理



```
MINGW64/d/android home/project/jiandao
$ git pull
Already up-to-date.

Admin@Admin-2Y MINGW64 /d/android home/project/jiandao (master)
$ git status
Auto-merging app/src/main/java/com/webbonn/jiandao/utils/JsonUtil.java
On branch master
Your branch is up-to-date with 'origin/master'.
Changes to be committed:
  (use "git reset HEAD <file>" to unstage)

    new file:   app/libs/gson-2.5.jar
    new file:   app/src/main/java/com/webbonn/jiandao/interfere/HttpCallBack.java
    new file:   app/src/main/java/com/webbonn/jiandao/ui/activity/UserChangePasswordActivity.java
    new file:   app/src/main/java/com/webbonn/jiandao/utils/Map2Json.java
    new file:   app/src/main/res/layout/activity_user_change_password.xml

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:  .idea.gradle.xml
        modified:  .idea/modules.xml
        modified:  Jiandao.iml
        modified:  app/app.iml
        modified:  app/build.gradle
        modified:  app/src/main/AndroidManifest.xml
        modified:  app/src/main/java/com/webbonn/jiandao/config/ServerConstant.java
        modified:  app/src/main/java/com/webbonn/jiandao/ui/activity/UserInfoActivity.java
        modified:  app/src/main/java/com/webbonn/jiandao/ui/activity/UserInfoEditActivity.java
        modified:  app/src/main/java/com/webbonn/jiandao/ui/activity/UserInfoSettingActivity.java
        modified:  app/src/main/java/com/webbonn/jiandao/utils/JsonUtil.java
        modified:  app/src/main/java/com/webbonn/jiandao/utils/NetUtil.java
        modified:  app/src/main/res/layout/activity_user_info.xml
        modified:  app/src/main/res/values/strings.xml
        modified:  gradle.properties

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        .gradle/
        build/
Dropped refs/stash@{0} (a756d56d6e077518fac0d8995ab8fdeb80ed054c)
Admin@Admin-2Y MINGW64 /d/android home/project/jiandao (master)
$
```

crifan.com, 使用署名4.0国际(CC BY 4.0)协议发布 all right reserved, powered by Gitbook最后更新: 2021-04-23 20:10:38

error RPC failed HTTP 504 curl 22

现象： Mac中用 SourceTree 去clone克隆代码结果出错：

```
git -c filter.lfs.smudge= -c filter.lfs.required=false -c diff.mnemonicprefix=
Cloning into '/Users/minglong/saicgroup_srt-ios'...
error: RPC failed; HTTP 504 curl 22 The requested URL returned error: 504 Gate
fatal: The remote end hung up unexpectedly
Completed with errors, see above
```

原因：git代码仓库中有大文件，网络不够好，始终无法下载下来，所以报错

解决办法：跳过大文件，此处该大文件在历史旧版本中，最新版中没有大文件。所以可以用只下载当前最新版的方式实现跳过旧版本，从而跳过大文件。

操作步骤：

```
git clone https://git.oschina.net/xxxxxx/xxx.git --depth=1
```

crifan.com，使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by
Gitbook最后更新：2021-04-23 20:11:25

git应用

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by
Gitbook最后更新: 2021-04-23 20:20:13

相关支持

git中文件的不同状态

git中，文件被改动后，会有多种不同的状态，往往用固定的字母去表示：

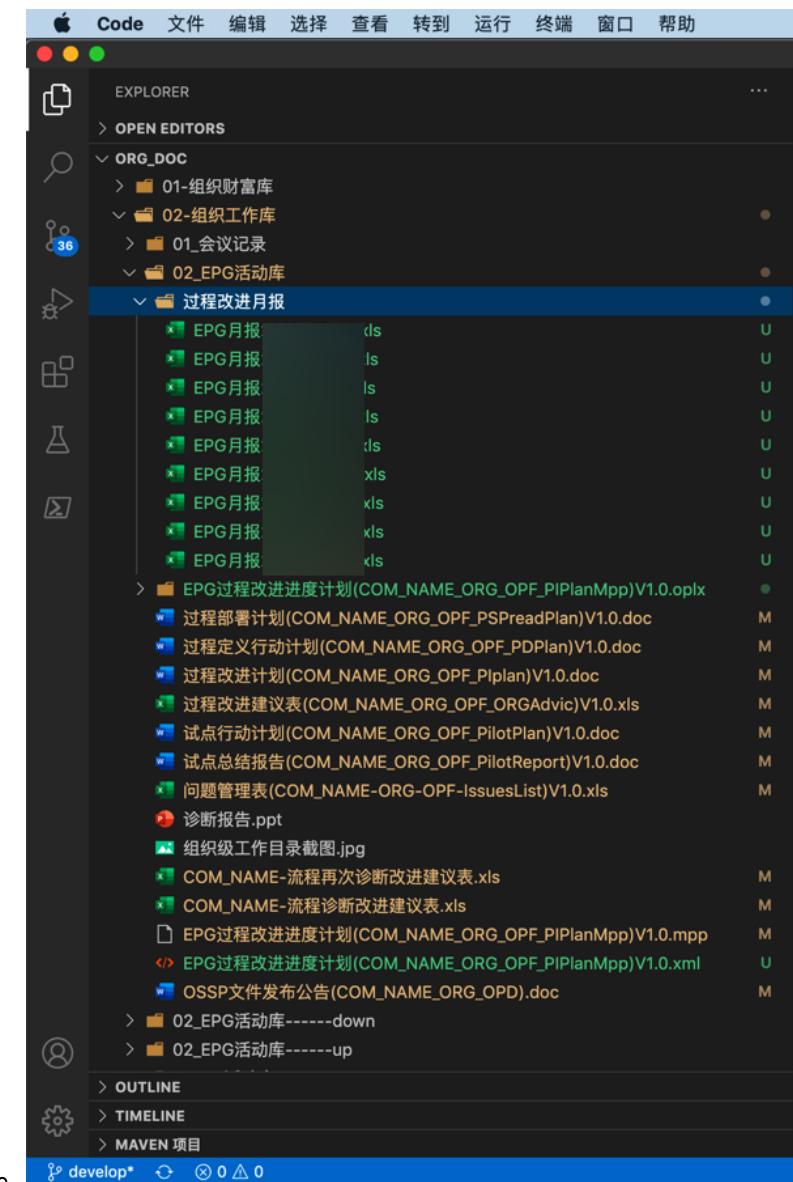
图标	含义
U	文件未追踪 (Untracked)
A	新文件 (Added, Staged)
M	文件有修改 (Modified)
+M	文件有修改 (Modified, Staged)
C	文件有冲突 (Conflict)
D	文件被删除 (Deleted)

对应的不同的支持git的工具中的效果：

VSCode

- U=文件未追踪

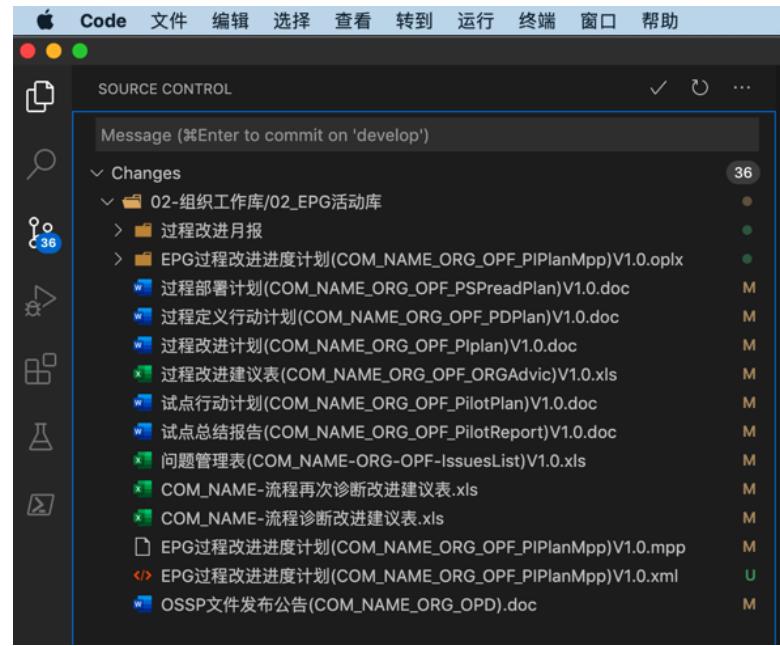
给git加代理



切换到git视图，且排序是按照目录树 的效果是：

- M = 文件有修改

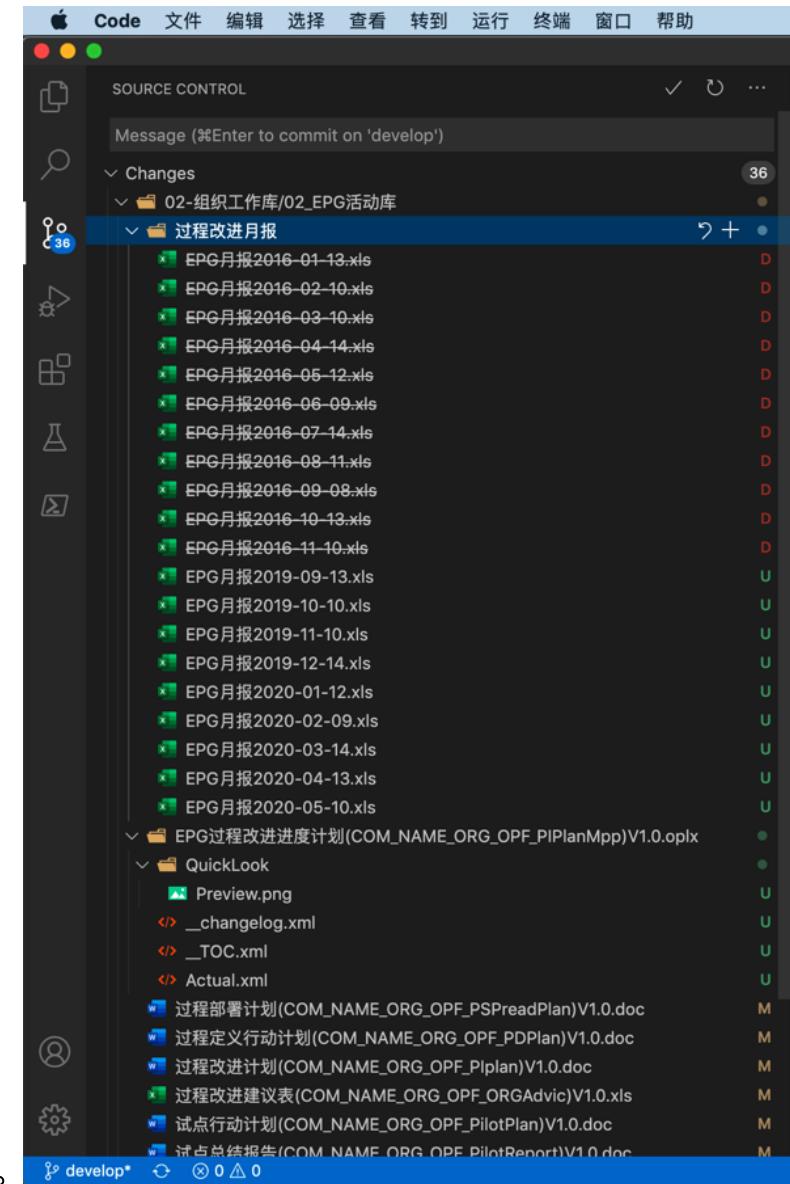
给git加代理



以及：

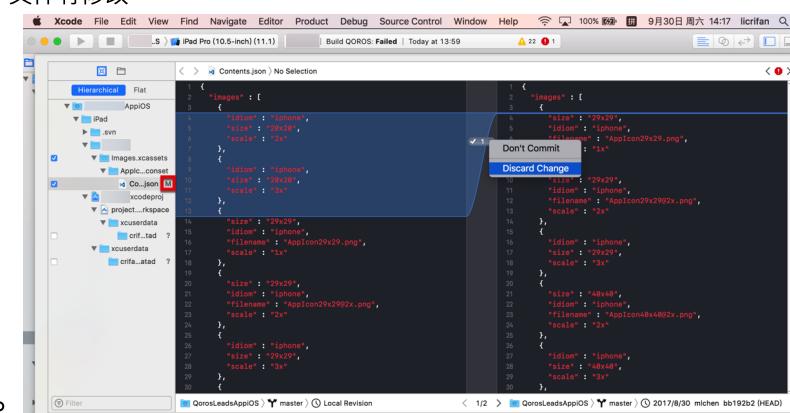
- D = 文件被删除

给git加代理



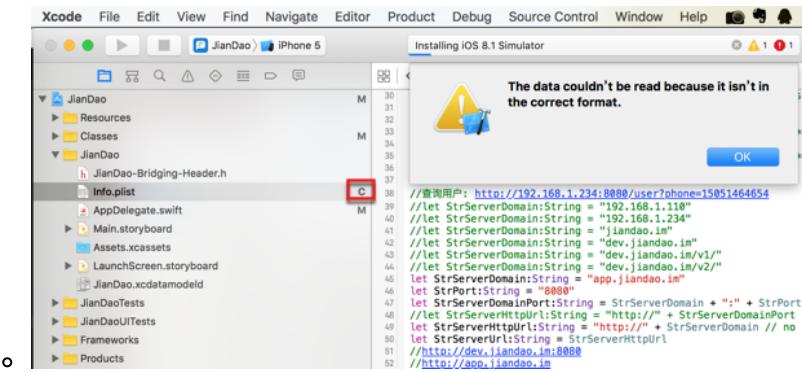
XCode

- M=文件有修改



- C=文件有冲突

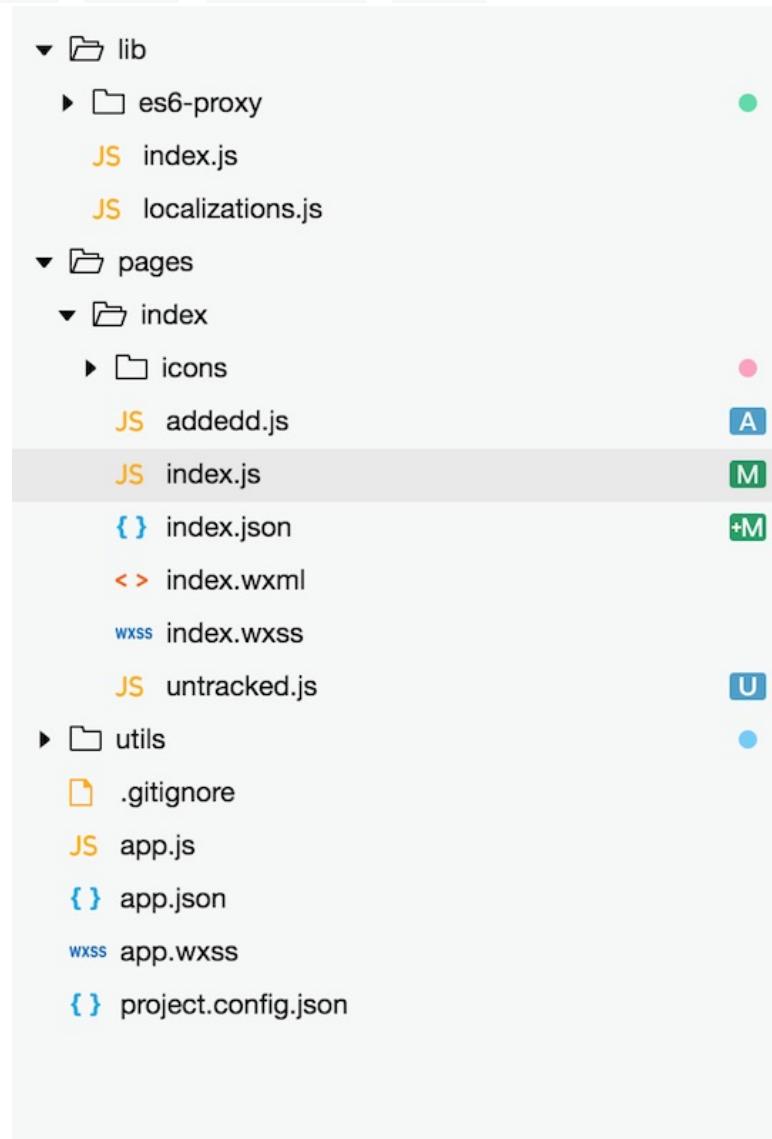
给git加代理



微信小程序开发工具

官网举例：

- A=新文件 + M=有修改 + +M=有修改且暂存 + U=未追踪



自己实例：

给git加代理

- M=有修改 + U=未追踪

The screenshot shows the WeChat DevTools developer tools interface. On the left, there's a preview window for an iPhone 5 device showing a simple UI with a profile picture and some text. To the right is the code editor for an app.js file. The code editor has several lines highlighted with colored underlines: blue for modified lines and green for untracked lines. A red box highlights one of these colored lines. Below the code editor is a console tab showing log output related to the 'onLaunch' event. At the bottom, there are tabs for 'Console', 'Sources', 'Network', 'Security', 'AppData', 'Audits', 'Sensor', 'Storage', 'Trace', and 'Wxmi'.

```
//app.js
App({
  onLaunch: function (options) {
    console.log("onLaunch: options=%o", options)
    // 展示本地存储能力
    var logs = wx.getStorageSync('logs') || []
    logs.unshift(Date.now())
    wx.setStorageSync('logs', logs)
  }
  // 登录
  wx.login({
    success: res => {
      // 发送 res.code 到后台换取 openId, sessionKey, unionId
    }
  })
  // 获取用户信息
  wx.getSetting({
    success: res => {
      if (res.authSetting['scope.userInfo']) {
        // 已经授权，可以直接调用 getUserInfo 获取头像昵称，不会弹框
        wx.getUserInfo({
          success: res => {
            App.globalData.userInfo = res.userInfo
            wx.setStorageSync('userInfo', res.userInfo)
          }
        })
      }
    }
  })
})
```

小程序中对应颜色表示额外信息：

图标	含义
小红点	目录下至少存在一个删除状态的文件
小橙点	目录下至少存在一个冲突状态的文件
小蓝点	目录下至少存在一个未追踪状态的文件
小绿点	目录下至少存在一个修改状态的文件

小程序代码

对应的小程序的代码中的带颜色的线条，表示对应的含义：

样式	含义
蓝色线条	此处的代码有变动
绿色线条	此处的代码是新增的
红色三角箭头	此处有代码被删除

举例：

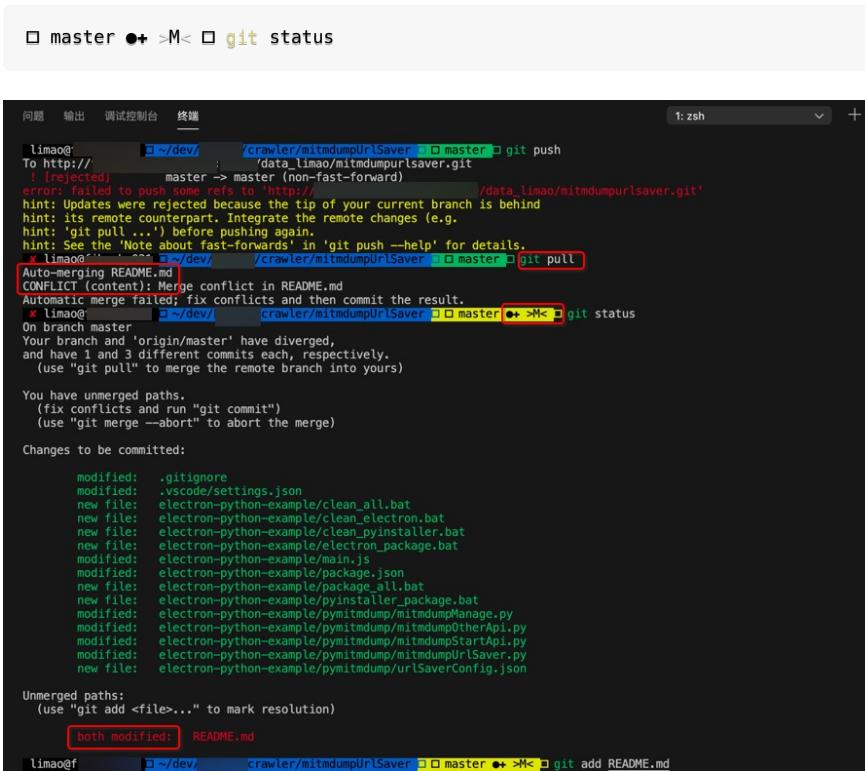
给git加代理

```
8     safeGet,
9     localized,
10    formatDate,
11    config,
12 } from '../../../../../lib/index.js'
13
14 // added
15 // added 2
16 import {
17   getAppInfo, // modified
18   getDownloadUrl,
19   abc,
20 } from '../../../../../utils/api.js'
21
22 import * as C from '../../../../../utils/config.js'
```

zsh

zsh 在安装了插件后，对于 git 支持的很好

甚至包括：当 auto-merge 出现 conflict 冲突时， git status 的前缀都自动显示出 >M<：



The screenshot shows a zsh terminal window titled '1: zsh'. The user has run 'git push' and received a rejection because their local branch is behind the remote. They then run 'git pull' to merge changes from the remote into their local branch. The output shows a conflict in the 'README.md' file. The terminal then displays 'git status' which includes the '>M<' prefix for files that have been modified during the merge process.

```
master •+ >M< git status
```

```
limao@... ~/dev/crawler/mitmdumpUrlsaver(master) git push
To http://.../data_limao/mitmdumpUrlsaver.git
! [rejected]      master --> master (non-fast-forward)
error: failed to push some refs to 'http://.../data_limao/mitmdumpUrlsaver.git'
hint: Updates were rejected because the tip of your current branch is behind
hint: its remote counterpart. Integrate the remote changes (e.g.
hint: 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
x limao@... ~/dev/crawler/mitmdumpUrlsaver(master) git pull
Auto-merging README.md
CONFLICT (content): Merge conflict in README.md
Automatic merge failed; fix conflicts and then commit the result.
x limao@... ~/dev/crawler/mitmdumpUrlsaver(master) git status
On branch master
Your branch and 'origin/master' have diverged,
and have 1 and 3 different commits each, respectively.
  (use "git pull" to merge the remote branch into yours)

You have unmerged paths.
  (fix conflicts and run "git commit")
  (use "git merge --abort" to abort the merge)

Changes to be committed:
  modified: .gitignore
  modified: .vscode/settings.json
  new file: electron-python-example/clean_all.bat
  new file: electron-python-example/clean_electron.bat
  new file: electron-python-example/clean_pyinstaller.bat
  new file: electron-python-example/electron_package.bat
  modified: electron-python-example/main.js
  modified: electron-python-example/package.json
  new file: electron-python-example/pymitmdump/all.bat
  new file: electron-python-example/pyinstaller_package.bat
  modified: electron-python-example/pymitmdump/mitmdumpManage.py
  modified: electron-python-example/pymitmdump/mitmdumpOtherApi.py
  modified: electron-python-example/pymitmdump/mitmdumpStartApi.py
  modified: electron-python-example/pymitmdump/mitmdumpUrlsaver.py
  new file: electron-python-example/pymitmdump/urlsaverConfig.json

Unmerged paths:
  (use "git add <file>..." to mark resolution)
    both modified: README.md
```

表示有内容需要合并（后才能再去提交）

此处表明，细节支持的很到位。

git的IDE

- 编辑器 或 IDE 本身集成Git功能
 - VSCode
- 专门的Git的客户端
 - Mac
 - SourceTree
 - Win
 - TortoiseGit
 - 官网: <https://tortoisegit.org/>



- 一句话描述: Windows Shell Interface to Git
- The Power of Git – in a Windows Shell

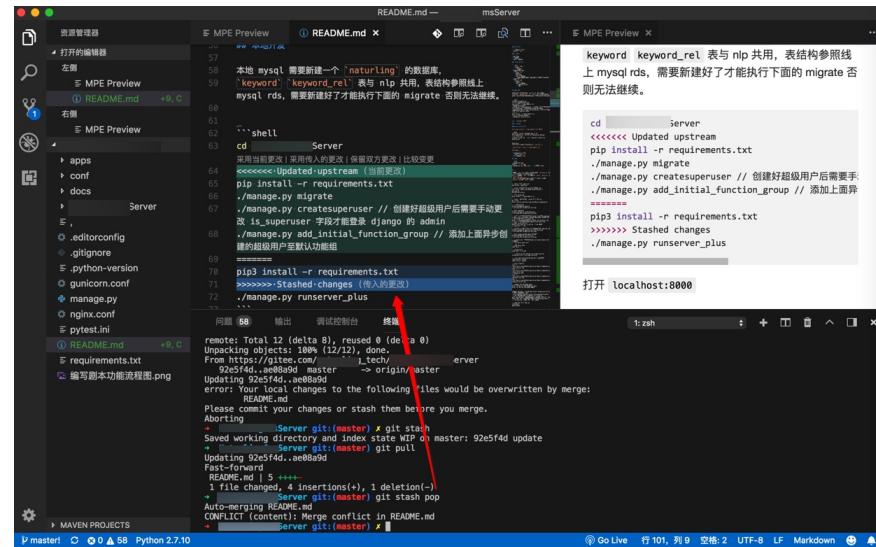
crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by
Gitbook最后更新: 2021-04-23 20:18:38

VSCode

VSCode本身自带功能，就默认支持git，且支持的非常好。

Conflict冲突

当（比如 git stash pop 后）出现conflict冲突后，会高亮显示冲突的地方：



且可以直接点击按钮实现对应效果：

- 采用当前更改
 - 即，使用： Updated upstream
- 采用传入的更改
 - 即，使用： Stashed changes
- 保留双方更改
 - 即，两者都保留
- 比较更改
 - 点击后可以比较内容差异

-> 方便你合并想要的内容，而不会导致内容丢失。

crifan.com，使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by
Gitbook最后更新：2021-04-23 20:19:14

在线git仓库

在线仓库，有很多。

比如：

- `github`
 - 网上用的最多的
 - 支持
 - 静态页面
 - [【已解决】用github的io去存放个人的静态页面](#)
- `gitee = 码云`
 - CSDN旗下的
 - 属于国内做的很不错的
 - 支持免费私有仓库的
 - 还支持企业版
 - [【记录】开通码云gitee的企业版](#)

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新：2021-04-23 20:20:09

基于git的系统

git如此好用，所以应用极其广泛。

后续出现很多基于git的系统：

- gitbook
 - 详见独立教程
 - [电子书制作利器：GitBook](#)

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by
Gitbook最后更新：2021-04-23 20:19:30

附录

下面列出相关参考资料。

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by
Gitbook最后更新: 2021-04-23 20:10:19

相关教程

- 不错的Git教程
 - 阮一峰的git教程
 - [Git远程操作详解 - 阮一峰的网络日志](#)
 - FFmpeg官网的git教程
 - [Using Git to develop FFmpeg](#)
 - Gitee码云的git教程
 - [码云 \(Gitee.com\) 帮助文档_V1.2](#)
 - [还有常见问题汇总](#)

The screenshot shows a mobile browser displaying the Gitee.com help document version 1.2. The title bar says "码云 (Gitee.com) 帮助文档 V1.2". Below it is a "FAQ" section with a sub-section titled "Git 操作管理". A red box highlights a list of common git issues:

- 新手小白如何快速的在码云平台注册账号并完成第一次提交
- http(s)方式如何自动记住密码
- 如何处理代码冲突
- 如何进行版本回退
- git submodule
- 为什么在push的时候，出现了413错误，push失败
- 为什么pull request合并不了
- 如何对一个项目提交Pull Request
- git clone git@osc项目出错 server aborted the ssl handshake
- 如何导入Github仓库到码云

- 其他一些git相关资料
 - [Git教程 - 廖雪峰的官方网站](#)
 - [git - the simple guide - no deep shit!](#)
 - [猴子都能懂的GIT入门 | 贝格乐 \(Backlog\)](#)
 - [Pro Git \(中文版\)](#)

参考资料

- 【已解决】给蝉大师搜索塔防结果的任务列表根据三国类关键字优先排序
- 【已解决】git中.gitignore中如何配置某文件夹下排除所有但只包含某子文件夹
- 【已解决】Gitlab仓库git pull报错fatal: unable to access Empty reply from server
- 【已解决】Gitlab中尝试用clone加mirror参数实现git仓库整体迁移且带历史提交日志
- 【已解决】Gitlab的旧git仓库迁移到新仓库且保留commit历史记录
- 【规避解决】Mac中给git添加加一次的当前的临时代理
- 【已解决】mac中git push只对github用代理而对gitee不用代理
- 【已解决】gitlab的仓库git push报错：fatal unable to access Empty reply from server
- 【已解决】git clone出错：Cloning into fatal Authentication failed for
- 【整理】git中PR Pull Request的含义和如何使用
-
- 【已解决】gitbook代码去git push --set-upstream origin master死掉没反应
- 【已解决】git stash pop出错：error Your local changes to the following files would be overwritten by merge
- 【已解决】git push出错：error failed to push some refs to
- 【已解决】git中创建新分支并设置默认提交到新分支
- 【已解决】如何用npm install去安装react-tappable的github中特定分支的代码
- 【已解决】github.io的git的push非常慢
- 【已解决】git pull出错：error Your local changes to the following files would be overwritten by merge
- 【已解决】git下载出错：warning templates not found /usr/local/git/share/git-core/templates – 在路上
- 【已解决】git和fabric中排除项目根目录下data文件夹而保留某子文件夹中data文件夹
- 【已解决】git记住密码不要每次都提示输入
- 【已解决】Xcode出错：The stickers icon set or app icon set named AppIcon did not have any applicable content – 在路上
- crifan/EvernoteToWordpress
- 【已解决】git下载代码出错：error RPC failed HTTP 504 curl 22 The requested URL returned error – 在路上
- 【已解决】git push出错：fatal: Authentication failed for – 在路上
- 【已解决】git去clone出错：fatal: Authentication failed for – 在路上
- 【记录】开通码云gitee的企业版
- 【已解决】用github的io去存放个人的静态页面
-
- 版本控制系统SVN、Git、Perforce区别 - 简书
- 【开发工具】最强Git使用总结 - pdai - 博客园
- github - Only use a proxy for certain git urls/domains? - Stack Overflow
- Git - git-config Documentation
- git 设置和取消代理
- 帮助 - Wiki - 码云 Gitee.com

- [如何导入外部Git仓库到中国源代码托管平台（Git@OSC） - 开源中国](#)
- [Using Git to develop FFmpeg](#)
- [代码编辑 · 小程序](#)
- [码云（Gitee.com）帮助文档_V1.2](#)
- [Git远程操作详解 - 阮一峰的网络日志](#)
- [Git教程 - 廖雪峰的官方网站](#)
- [git - the simple guide - no deep shit!](#)
- [猴子都能懂的GIT入门 | 贝格乐（Backlog）](#)
- [Pro Git（中文版）](#)
-

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by
Gitbook最后更新: 2021-04-23 20:10:24