

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
常见问题	1.6
unable to access Empty reply from server	1.6.1
git应用	1.7
相关支持	1.7.1
git的IDE	1.7.2
在线git仓库	1.7.3
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： `v0.4`
- 更新时间： `20210421`

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，及其相关的配置文件，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；以及整理了一些常见问题；整理出git相关应用，相关的支持、git的IDE、在线的git仓库系统等。最后给出相关教程。

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

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。如有发现侵犯您的版权，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： 2021-04-21 20:51:30

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	—

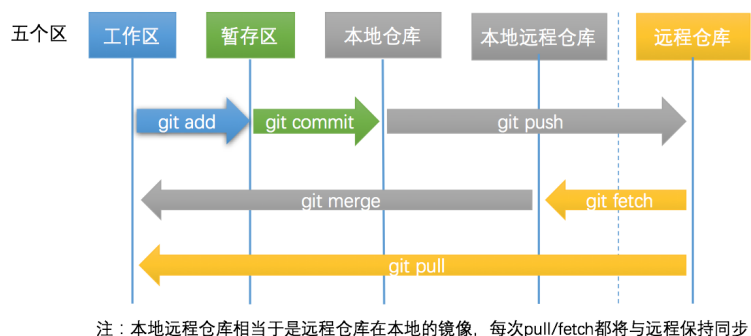
crifan.com，使用署名4.0国际(CC BY 4.0)协议发布 all right reserved，powered by Gitbook最后更新：2021-04-21 20:36:46

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
```

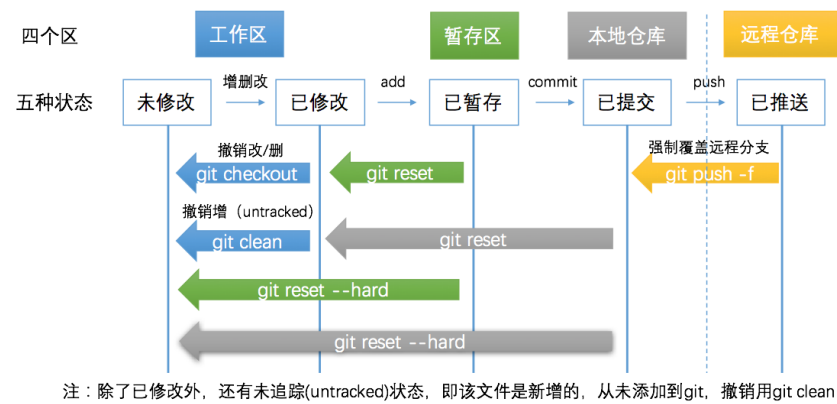
- 或：直接1步

- 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
```

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: 2021-04-21 20:36:08

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: 2021-04-21 20:37:03

.gitignore

- .gitignore
 - 是什么：一个普通文件
 - 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

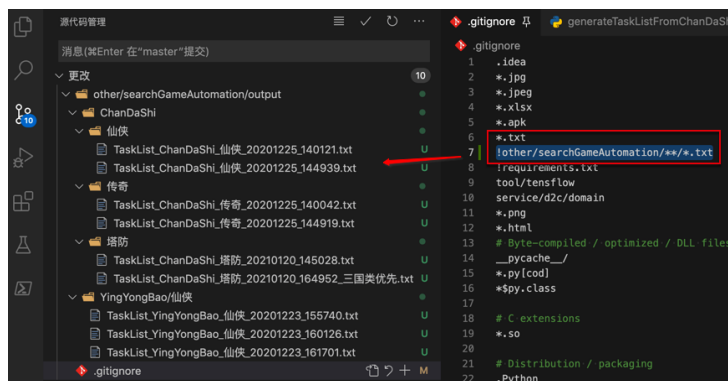
常见问题

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
```

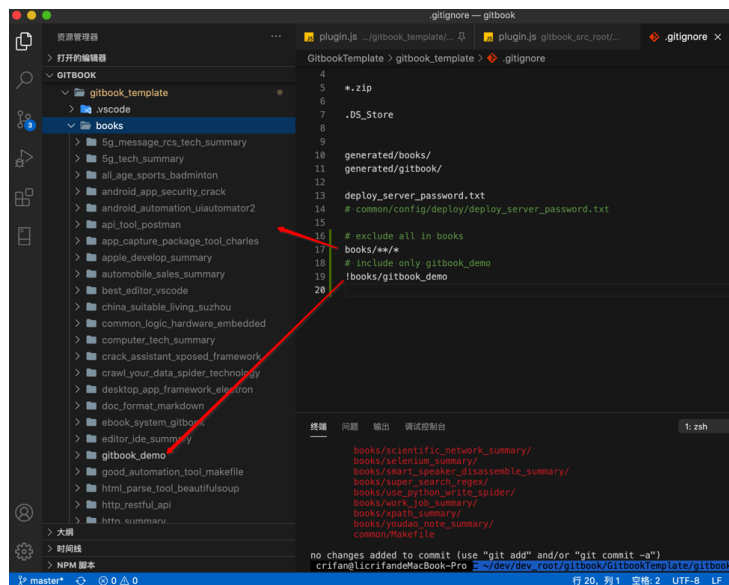
- 效果



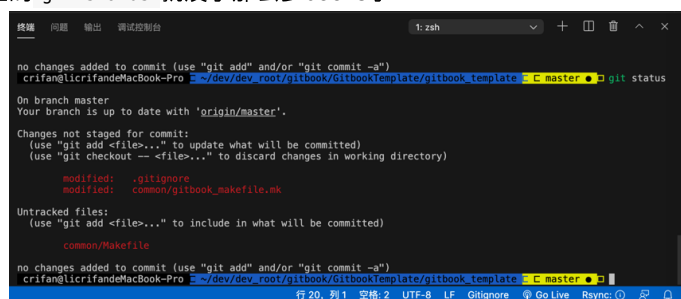
- 举例2
 - 希望：排除掉books下面所有的子文件夹
 - 但是只保留books/gitbook_demo
 - .gitignore 的写法

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

- 效果



- 相应的 git status 就没了那么多books了



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

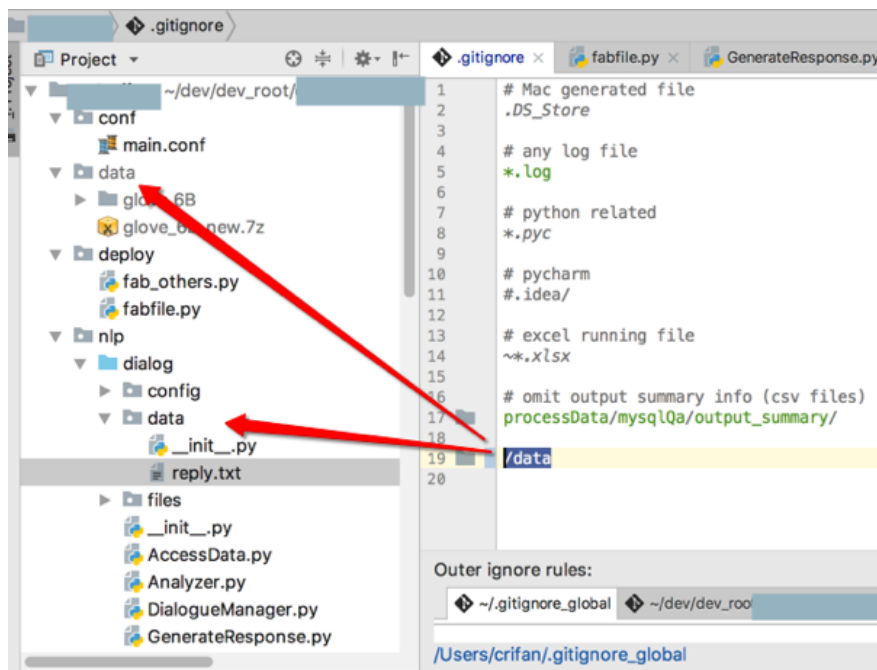
.gitignore 的配置：

```
/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

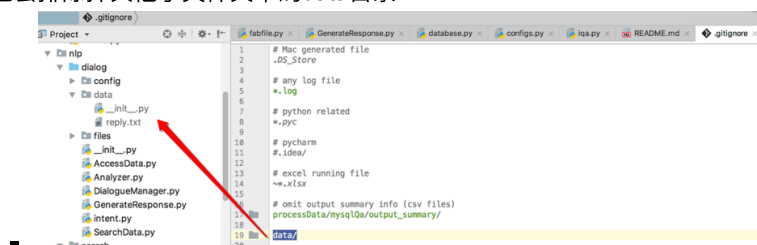
效果：



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

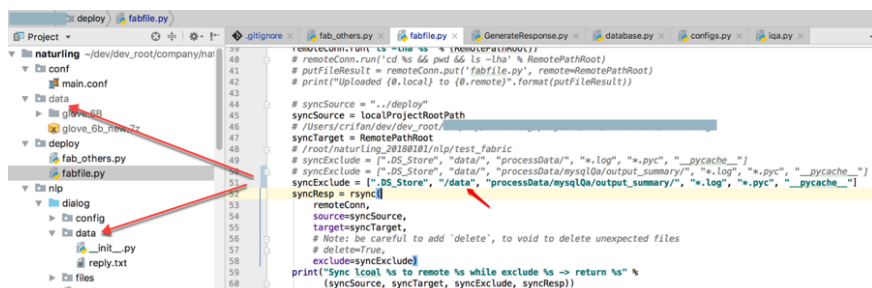


另外：

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: 2021-04-21 20:37:01

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

- 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: 2021-04-21 20:36:58

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。

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

```
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

```
[http]
```

- 方式2: 设置 proxy 值是空

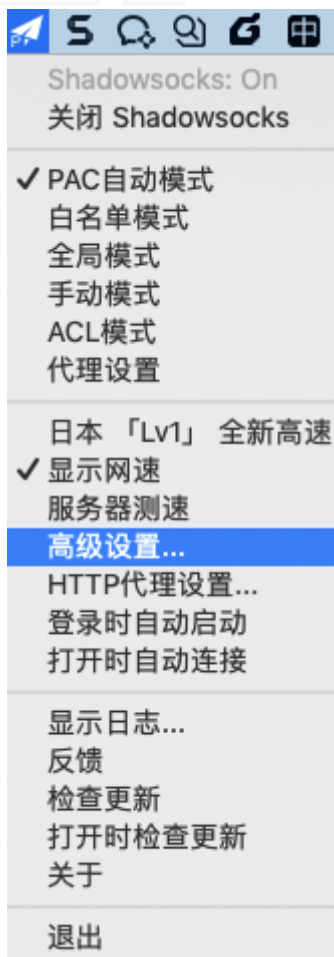
```
[http]  
proxy =
```

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Socks5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：

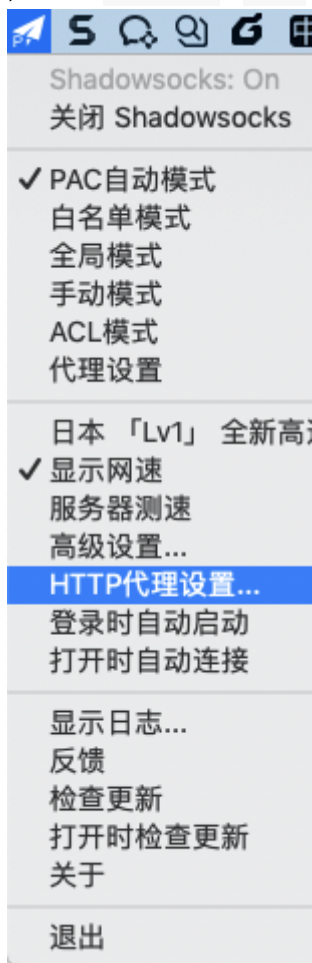
- socks5://127.0.0.1:1086

- （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

- 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://127.0.0.1:1086
```

- 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

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

- 全局

```
git config --global http.proxy socks5://127.0.0.1:1086
git config --global http.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: 2021-04-21 20:36:51

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: 2021-04-21 20:36:29

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/naturling_crifan/see_empty_project_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/naturling_crifan/see_empty_project_git
git push -u origin master
```

举例：EvernoteToWordpress

[crifan/EvernoteToWordpress](#)



The screenshot shows the Gitee web interface for the repository `crifan/EvernoteToWordpress`. The page includes a header with the Gitee logo and navigation links. The main content area displays the README file, which provides instructions for setting up the repository. The README text includes a quick setup section with a direct link to the repository, a section for repository recommendations (README, LICENSE, .gitignore), a link to the Git入门? page, a link to the Git 全局设置: page, a section for creating a git repository with a code block of commands, and a section for existing repositories with another code block of commands.

```
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/EvernoteToWordpress

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: 2021-04-21 20:36:19

记住密码

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

之后，正常git操作，比如：

```
git pull
```

输入账号和密码

-> git就会记住了

-> 下次再git操作，就不要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： 2021-04-21 20:36:32

迁移仓库且保留历史记录

之前遇到过个需求：迁移仓库 整体迁移 且保留所有历史提交记录

想要把git仓库，整体迁移，且保留全部历史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：

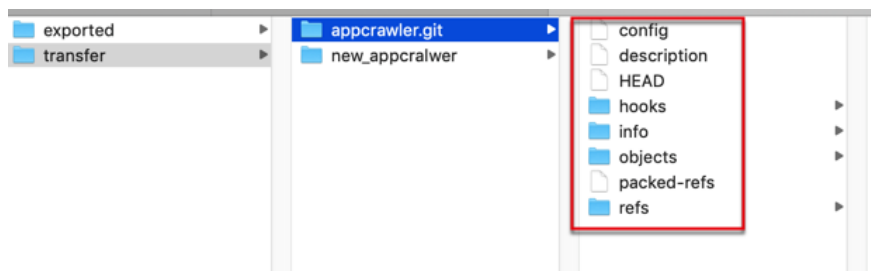
以镜像方式下载复制代码

```
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

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://fibodtidc.corp.com:12310/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

```
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： 2021-04-21 20:36:26

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: 2021-04-21 20:36:15

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: 2021-04-21 20:36:12

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: 2021-04-21 20:36:42

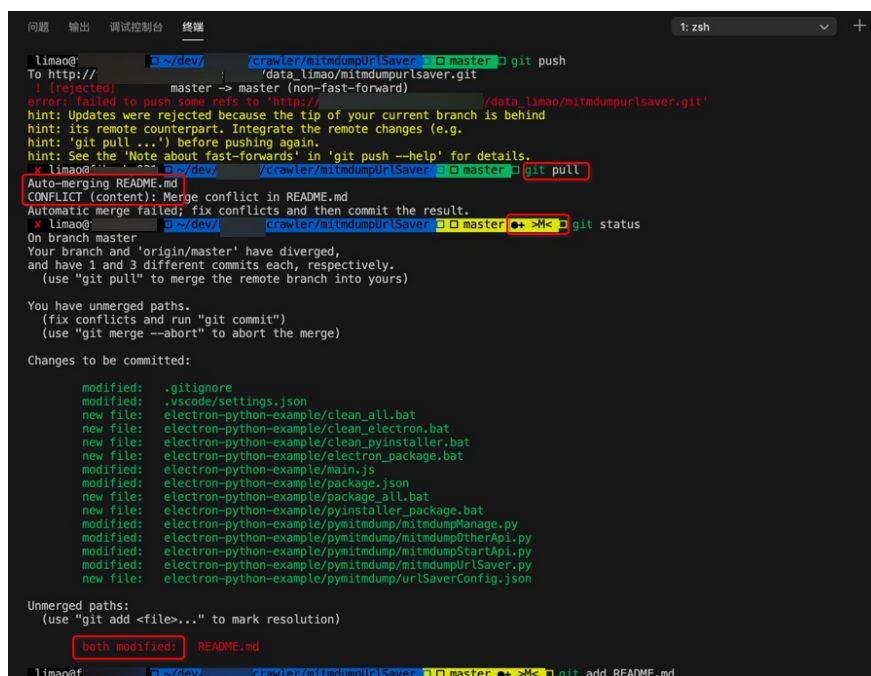
相关支持

zsh中对于git支持的很好

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<:

```
□ master ●+ >M< □ git status
```



The screenshot shows a terminal window with a dark background. The prompt is 'limao@'. The user has run 'git push' and received an error: 'error: failed to push some refs to http://.../data/limao/mitmdumpurl saver.git'. The error message suggests that updates were rejected because the tip of the current branch is behind its remote counterpart. The user then runs 'git pull', which triggers an 'Auto-merging README.md' process. A conflict is detected in 'README.md'. The terminal shows the conflict resolution process, including a list of changes to be committed and a list of unmerged paths. The 'git status' command is run again, showing the conflict markers '>M<' for the 'README.md' file. The terminal also shows the file structure of the project, including 'electron-python-example' and 'pymitdump' directories.

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。

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：2021-04-21 20:36:44

git的IDE

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: 2021-04-21 20:36:35

在线git仓库

- github
- gitee = 码云

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: 2021-04-21 20:36:39

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: 2021-04-21 20:35:53

相关教程

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: 2021-04-21 20:36:02

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](#)
- [crifan/EvernoteToWordpress](#)
-
- [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](#)
-

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: 2021-04-21 20:35:58