# Git补完计划

```
/*  @file Git补完计划.md
*   @author KurehaTian
*   @date 2022-02-02
*   Copyleft (c) 2022 Cyber10@Cybersmartcar
*/
```

# 零、引言

请读者善用浏览器的 **索引目录** 功能。

本文意在快速指导读者掌握使用Git进行版本控制与多机同步。

> 版本控制
>
> 在任务不同阶段，对项目文件进行存档管理
>
> 高技术力：使用Git进行更新维护，每次更新自动上传与上次不同的部分，随时可以回退
>
> 低技术力：在钉钉群里打成一个一个压缩包，每次都要整个重新下载，版本切换要建立许多文件夹
>
> e.g.环岛未完善.zip、交流赛代码.zip、陀螺仪卡初始化.zip、环岛又好了啊啊啊啊啊.zip、加角速度环牛逼格拉斯.zip、全村的希望.zip、环岛又好了.zip [引自16届智能车某组]

> 多机协同
>
> 用两台或以上电脑写代码，在实验室写不完的带回宿舍在另一台电脑上继续写。
>
> 高技术力：写之前从Git抓取最新版本，写完之后用Git推送
>
> 低技术力：用U盘拷过来拷过去，用钉钉/QQ传，最后找不到改动了哪里

此外，Git还能实现多人协同，分支管理等操作，限于篇幅在此不做讨论。

```
参考文献：
https://www.liaoxuefeng.com/wiki/896043488029600
https://www.runoob.com/git/git-tutorial.html
```

---

# 一、准备工作

## 1. 安装Git

```
下载链接：https://git-scm.com/
```

安装大体可以一路Next，默认编辑器选建议Vim，可以把Git添加到PATH

安装完成后，在开始中找到 Git>Git Bash，会蹦出一个类似于CMD命令行的窗口。

在里面输入：

```
git config --global user.name "你的用户名"
git config --global user.email "你的邮箱地址"
```

## 2. 注册Gitee账号

```
网址：https://gitee.com/
```

相信这一步应该不用教吧。

## 3.给账号添加数字签名

这一步是为了帮助远程托管平台确认本机身份。

打开Git Bash,输入：

```
ssh-keygen -t rsa -C "你的邮箱"
```

中间会弹出一系列询问(2次)，直接按回车即可。

去C:/Users/你的用户名/.ssh/里应该会看到

> id_rsa.pub
>
> id_rsa

两个文件，分别是你的公钥和私钥，请务必不要泄露你的私钥。

用记事本打开*id_rsa.pub*，复制里面的内容。

在Gitee主页，右上角用户头像 -> 菜单“修改资料”，然后选择“SSH公钥”，填写一个便于识别的标题，然后把用户主目录下的.ssh/id_rsa.pub文件的内容粘贴进去。

*如果你使用了多台电脑，那么每台电脑都需要添加数字签名*

> 这几个概念可能会引起混淆，在此进行解释。
>
> Git:用于进行管理的版本控制系统，在本地运行，也可以与托管平台进行联动
>
> Gitee/Github：Gitee是国内较大的一个代码托管平台，Github是全球使用人数较多的托管平台，但由于在境内没有服务器，所以访问速度较慢。
>
> 打个不太恰当的比方，Git是你手里的手机，Gitee/Github就相当于中国联通/中国电信等运营商，你拿着手机当然可以自娱自乐，但你要接入运营商的网络才能与别人沟通或者在自己的多台手机之间传消息。
>
> 而如果你只去运营商开通了号码拿到了卡，却没有手机，就自然没法通信了。不过你也可以用这个账号去打公用电话(直接在网页上下载上传,但失去便利性与网盘无异)

---

# 二、新建Gitee项目库

登录gitee后,再直接访问Gitee的主页看到的页面是"我的工作台"。鼠标移至页面右上角的“+”后出现菜单，选择“新建仓库”。

仓库名称随意起，不过鉴于以后经常会跟这个路径打交道，在此建议：

> 1.不要太长，不然每次手敲这个名字的时候你会后悔的
>
> 2.不要包含中文，否则以后可能某些软件会不兼容。

写好仓库名称之后，路径会自动生成的。

如果你不是想开源，让所有人都看到代码的话，就选择私有好了。

建议下面的初始化仓库、设置模板、选择分支模型都打上勾：

> 初始化仓库打勾即可，底下三个什么都不用选
>
> 设置模板选择后底下选择Readme文件
>
> 选择分支模型打勾后选择单分支模型(只创建Master分支)

---

# 三、克隆Gitee项目到本地

还记得你在Gitee上新建的仓库吗？在他的主页里有一个显眼的“克隆/下载”按钮，点一下进去后选择"SSH"选项卡，然后把下面的链接复制下来。

比如新建的就叫做

> git@gitee.com:kurehatian/general-drone-frame.git

在你的硬盘上找一个风水宝地。

比如说，我打算把一个叫做general-drone-frame的项目添加到G:\general-drone-frame（这个文件夹就是项目文件夹),

那么在“计算机”里面找到你要放项目的**上一级目录。**

右键,选择"Git Bash Here".熟悉的终端窗口又出现了。

那么在Git Bash里输入：

```
git clone 你的链接
```

比如我的话就是

```
git clone git@gitee.com:kurehatian/general-drone-frame.git
```

他很快就会把项目里的文件夹下载到你本地，因为确实没有啥好下的。

> 远程库与本地仓库并不是时时刻刻同步的，需要进行手动同步。
>
> 又到了举例子时间：你在项目里所做的编辑相当于在草稿纸(工作区)上解题，把你不是很确定的答案先写在试卷(暂存区)上，检查之后把最终结果写在答题卡(本地仓库)上，最后要把答题卡交给老师(远程仓库)。

---

# 四、把你的工程扔到本地仓库里

拿到了空白的试卷和答题卡，你要开始解题了。

对于我们已经有一个项目框架而言，要先把项目的基础部分跟本地仓库和远程仓库同步起来，这样版本管理器才能追踪你所做的更改。

## 1.写一下Readme

把你的项目扔到刚才你从远程仓库抓取的空仓库里，然后可以根据需要编辑一下README.md,这是你从网页上访问你仓库的门面，可以美美的装饰一下。具体里面怎么写，可以去看看Markdown的写法。

## 2.gitignore的配置

.gitignore里面写你不想进行同步的文件，比如每次编译后obj文件夹里产生的一堆.o之类的过程性文件没有同步的必要，不仅会浪费大量网络流量、很快的把你并不是很多的空间容量占满，更会让你查看日志时找不到每次更新的重点。

那么.gitignore要怎么写呢?

还是以刚才的项目为例，假如现在的文件目录是这样的:

```
G:\general-drone-frame\
    |-RT1064
    |   |-Project
    |   |   |-MDK
    |   |   |    |-Objects
    |   |   |    |   |-这里面都是我不想更新的文件
    ...其他文件先不用管，总之就是看这个Objects里的文件不顺眼。
```

那么我需要在.gitignore里写:

```
RT1064/Project/MDK/Objects/
```

表示这个文件夹里的所有文件不被同步。

如果有使用VScode的同学，记得要把.vscode文件夹也加入.gitignore。

如果还有其他不想同步的文件夹(比如测试数据等)，也可以直接往下继续写，从项目根目录开始的相对目录即可。

**切记：如果一个文件/文件夹一旦被同步过，那么之后即使加入.gitignore也没有用了，所以哪些文件要被忽略一定要在一开始就确定好。**

---

**如果你使用VScode写代码，其实是有一些可以不用自己手敲命令的手段的。具体操作可以参考这篇文章**

> https://blog.csdn.net/qq_qian/article/details/107384191

---

# 五、将工作区内容保存至暂存区

> git add [文件名]

通常为了避免遗忘某些更新了的文件不被同步，可以直接

> git add .

来把所有发生了变化的文件添加到暂存区。

如果你要将某个文件从暂存区移除，可以使用

> git rm --cached [文件名]

如果要清除整个暂存区，可以使用

> rm .git/index

---

# 六、将暂存区的更新提交到仓库

> git commit -m "引号里写本次提交的备忘，一定要这样写，不然还是会弹出来文本编辑器让你写的"

做到这一步的话就算是可以在本地实现版本管理了，实现版本控制已经足够了。如果还要做到多机协同的话，需要推送到远程仓库。

---

# 七、把本地仓库推送到远程

> 第一次推送时带上-u，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
>
> git push -u origin master
>
> 以后推送就是
>
> git push  origin master

---

# 八、从远程仓库拉取最新版本

```
git pull origin master
```

切记每次在与上次工作的电脑不同时一定要执行这一步，不然两个修改过的文件合并时可能会造成冲突。

---

# 九、版本回退

如果你这个版本写崩了，可以考虑退回以前的版本

```
//退回前一个版本
git reset --hard HEAD^

//退回前两个版本
git reset --hard HEAD^^

//退回前10个版本
git reset --hard HEAD~10
```

---

# 十、删除文件

还记得上面的文件目录吗？这次我们要删除Project文件夹了。

```
cd RT1064           //切换当前目录进入RT1064文件夹
git rm -f Project   //删除Project文件夹
cd ..               //切换到上一级文件夹，即G:\general-drone-frame
```

这条指令也适用于删除单个文件

具体用法请参考

https://www.runoob.com/git/git-rm.html

---

以上是基础内容的简介，下面给出便于日常使用的速查表。(适用于单分支情况)

---

# 速查表

以下命令均在项目文件夹执行

## 暂存区操作

| 功能 | 指令 |
| --- | --- |
| 把工作区的所有更改添加到暂存区 | git add . |
| 删除暂存区某个文件 | git rm --cached [文件名] |
| 清空暂存区 | rm .git/index |

## 本地仓库操作

| 功能 | 指令 |
| --- | --- |
| 将暂存区更改提交到本地仓库 | git commit -m "更新备注" |
| 删除文件 | git rm |
| 退回前一个版本 | git reset --hard HEAD^ |
| 退回前两个版本 | git reset --hard HEAD^^ |
| 退回前10个版本 | git reset --hard HEAD~10 |

## 远程仓库操作

| 功能 | 指令 |
| --- | --- |
| 从远程仓库拉取最新版本 | git pull origin master |
| 将当前本地仓库推送到远程仓库 | git push origin master |

## 工作流程

> 云端备份：
>
> 1.从远程仓库拉取最新代码： git pull origin master
>
> 2.开始编辑代码(像往常一样)
>
> 如果要删除文件的话，要在Git Bash 中使用git rm 来让版本管理系统知悉，否则远程仓库中对应文件不会被删除
>
> 如果要回退到以前的版本，当前没有添加到暂存区的所有操作将会丢失。(添加到暂存区的也会:P)
>
> 3.将所有更改添加到暂存区: git add .
>
> 4.将暂存区提交到本地仓库: git commit -m "备注"
>
> 5.将本地仓库推送到远程仓库: git push origin master

> 仅本地用作版本管理：
>
> 上述步骤的第2.到第4.步

**所有命令的详细用法请参考[https://www.runoob.com/git/git-basic-operations.html](https://www.runoob.com/git/git-basic-operations.html)**

**或者查看附件的手册**