注意

文章里面出现的

所有”口”代表“空格”

所有“紫色字体”代表“代码”

所有“绿色字体”代表“可替代文字”

# 一、安装Git

* 安装git什么的基本都是傻瓜式安装，不做讲解。

# 二、 检查你的Git版本

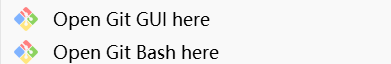
* win+R
* 输入cmd
* 输入git□--version

# 三、Git架构

# 四、Git基本使用

**4.1. 创建版本库**

* 在工作空间的目录中，右键“Git Bash Here”打开git终端

****

* 在Git终端中输入git□init指令，创建版本库(你的文件夹下会多出一个具有“隐藏”属性的“.git”文件)

**4.2查看版本库的状态**

|  |
| --- |
| git□status |

**4.3将工作空间的修改添加到**

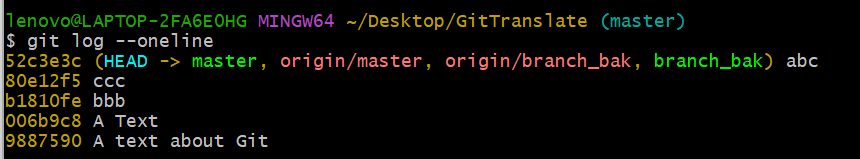
|  |
| --- |
| git□add□<文件名> //将指定文件add到暂存区 |
| git□add□. //将工作空间中所有文件add到暂存区 |

**4.4将暂存区内容提交到版本库(仓库)**

|  |
| --- |
| Git□commit□-m ‘版本说明’ |

**4.5查看历史版本库中的历史版本**

|  |
| --- |
| git□log --oneline //每个版本信息只显示一行 |
| git□log //显示每个版本的详细信息 |

其中

HEAD表示最新版本库

origin 表示云端最新版本库

**4.6设置用户信息(建议这里格式尽量保持和GitHub的信息保持一致)**

|  |
| --- |
| 因为我们将暂存区的内容提交到版本时，会记录当前版本的提交的用户信息，因此在版本提交之前需要先绑定用户信息 |

|  |
| --- |
| git□config --global□user.name□‘用户名’ |
| git□config --global□user.email□’电子邮箱’ |

完成一次后就不用反复进行此操作

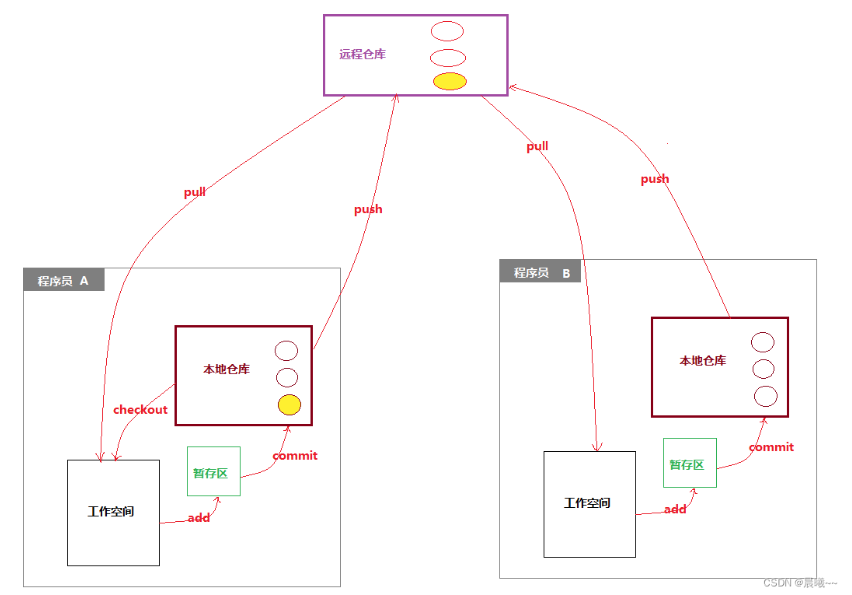
**4.7同步历史版本到工作空间**

|  |
| --- |
| git□checkout□版本号 |

# 五、 远程仓库

**5.1远程仓库**

|  |
| --- |
| 远程仓库，远程版本库；实现版本库的远程存储，以实现团队的协同开发 |

**5.2如何获取远程仓库**

可以使用GitHub创建仓库(当然还有许多仓库提供商),如何使用Github以及如何创建仓库这里不做详细讲解

仓库提供商

|  |
| --- |
| GitHub https://gitbub.com |
| Gitee（码云） https://gitee.com |
| Coding |

# 六、 远程仓库操作

**6.1准备工作(这里使用GitHub)**

1..创建一个文件夹当做本地仓库

2.初始化本地仓库

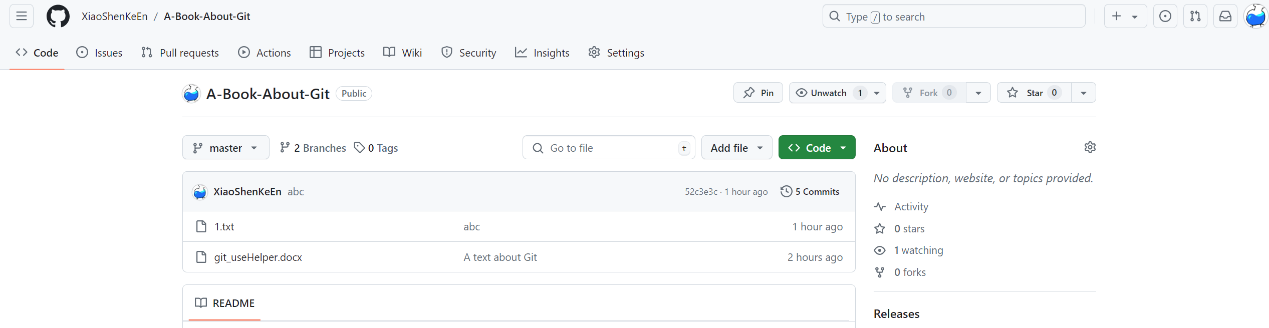
|  |
| --- |
| git□init |

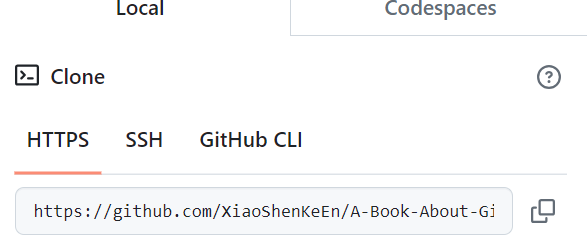
3.将工作空间搭建的项目结构add到暂存区

|  |
| --- |
| git□add□. |

4.将暂存区文件提交到版本库，生成第一个版本

|  |
| --- |
| git□commit□-m□’版本说明’ |

5.然后去GitHub找到你的远程仓库地址

点击<>Code

这个就是你的远程仓库地址，注意是HTTPS

**6.2.1本地仓库关联远程仓库**

|  |
| --- |
| git□remote□add□origin□http://\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |

**6.2.2查看远程仓库状态**

|  |
| --- |
| git□remote□-v |

**6.3 将本地仓库push到远程仓库**

|  |
| --- |
| git□push□origin□master //这里master是我的主分支，可以替换为其他分支的名称 |

**6.4其他开发者pull远程仓库到本地**

**6.4.1**

创建一个文件夹，并打开Git终端

创建本地仓库

|  |
| --- |
| git□init |

拉取远程仓库到本地

|  |
| --- |
| git□pull□[http://\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*](http://***************)□master //同理，这里master可以替换 |

**6.5解决开发冲突问题**

小X和小Y都是1.txt这个项目的开发人员，他们同时将1.txt pull到了各自的本地

* 小X在1.txt中添加了”aaa”
* 小Y在1.txt中添加了“bbb”
* 小Y已经将修改push到了Github上了，这时小X再push到GitHub上，这时小X会报错

如何解决此问题？

|  |
| --- |
| git□pull□https://\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*□master //先将小Y的pull下来 |

这时文件内部会出现这种格式，对文件进行冲突合并即可

|  |
| --- |
| >>>>>>>>>>>>>>>>>>>>>HEAD //最早修改  aaa  =================  bbb //你的修改  <<<<<<<<<<<<<<<<<<<<< sikdfhjkasdfhjasdfhjk |

然后再上传

|  |
| --- |
| git□add□. |
| git□commit□-m ‘版本名称’ |
| git□push□https://\*\*\*\*\*\*\*\*\*\*\*\*\*□master |

# 七、 分支管理

**7.1 什么是分支**

|  |
| --- |
| 分支就是版本库中记录版本位置（支线），分支之间项目会影响，使用分支可以对项目起到保护作用  分支就是一条时间线，每次提交就在这条时间线上形成一个版本 |

**7.2 分支特性**

* 创建一个新的版本库，默认创建一个主分支—master分支
* 每个分支可以进行单独管理（常规分支、保护分支、只读分支）
* 分支是可以合并的

**7.3分支操作**

**7.3.1 创建分支**

|  |
| --- |
| git□branch□<分支名称> |

**7.3.2 查看分支**

|  |
| --- |
| git□branch |

**7.3.3 切换分支**

|  |
| --- |
| git□checkout□<分支名称> |

**7.3.4 检出分支**

|  |
| --- |
| git□checkout□<历史版本>□-b□<分支名称> |

**7.3.5合并分支**

首先切换到你要**合并到的分支上(例如我要将master合并到branch\_bak,那么我就切换branch\_bak)**然后

|  |
| --- |
| git merge master |

**7.4将分支上传到GitHub上(前面创建的分支都仍存在本地)**

切换到你要上传的分支，然后

|  |
| --- |
| git□push□origin□<分支名称> |

确认分支是否上传到github上

|  |
| --- |
| git□branch□-r |