Git和Github学习笔记

1. Git是什么?

为了取代BitKeeper来管理Linux内核

基于命令行的版本控制工具

Git是目前世界上最先进的分布式版本控制系统（没有之一）。

2.安装Git

可以在Linux、windows、Mac OSA安装

windows在官网直接下载然后默认安装，从开始找到Git->git bash

设置名字和邮箱

```
$ git config --global user.name "Your Name"
$ git config --global user.email "email@example.com"
```

创建版本库

```
$ mkdir learngit
$ cd learngit
$ pwd
```

把目录变为可管理的

```
$ git init
```

编写readme.txt

```
vi readme.txt
```

把文件添加到仓库

```
$ git add readme.txt
```

把文件提交到仓库

```
$ git commit -m "wrote a readme file"
```

3.版本回退

掌握工作区的状态

```
$ git status
```

查看版本difference

```
$ git diff readme.txt
```

查看提交日志

```
$ git log
$ git log --pretty=oneline
```

回退版本

```
$ git reset --hard HEAD^
```

记录每次命令

```
$ git reflog
```

4.撤销修改

场景1：当你改乱了工作区某个文件的内容，想直接丢弃工作区的修改时，用命令`git checkout -- file`。（相当于草稿丢弃）

场景2：当你不但改乱了工作区某个文件的内容，还添加到了暂存区时，想丢弃修改，分两步，第一步用命令`git reset HEAD <file>`，就回到了场景1，第二步按场景1操作。（把add进暂存区的丢弃）

场景3：已经提交了不合适的修改到版本库时，想要撤销本次提交，参考版本回退一节，不过前提是没有推送到远程库。（把commit进板块的丢弃）

5.删除文件

文件管理器删除

```
$ rm test.txt
```

版本库删除

```
$ git rm test.txt
$ git commit -m "remove test.txt"
```

删错了还原

```
$ git checkout -- test.txt
```

(`git checkout`其实是用版本库里的版本替换工作区的版本，无论工作区是修改还是删除，都可以“一键还原”。)

6.远程仓库

Git和Github之间传输通过ssh加密

第1步：创建SSH Key。Windows下打开Git Bash创建SSH Key：

```
$ ssh-keygen -t rsa -C "youremail@example.com"
```

在用户主目录里找到`.ssh`目录，里面有`id_rsa`和`id_rsa.pub`两个文件，这两个就是SSH Key的秘钥对，`id_rsa`是私钥，不能泄露出去，`id_rsa.pub`是公钥

第2步：登陆GitHub，打开“Account settings”，“SSH Keys”页面，在Key文本框里粘贴`id_rsa.pub`文件的内容

- 添加远程库

在GIthub上创建一个新仓库

在本地文件夹下运行

```
$ git remote add origin git@github.com:name/learngit.git
```

把本地内容推送到远程库上

```
$ git push -u origin master
$ git push origin master
```

!其中遇到了git查找的remote名字和github不一样，报错fatal: Could not read from remote repository.

将本地库的所有内容推送到远程库，输入命令：

```
$ git remote add origin git@github.com:Alreadgo/usegit.git
```

```
$ git push -u origin master
```

git bash报错：

```
ERROR: Repository not found.
fatal: Could not read from remote repository.
```

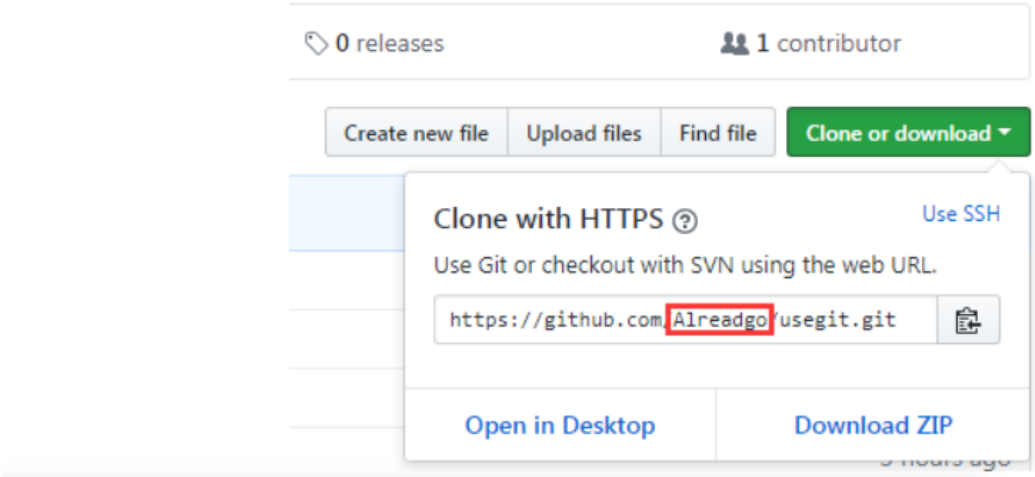
输入命令：

```
$ git remote -v
```

git bash输出：

```
origin  git@github.com:Nanna/usegit.git (fetch)
origin  git@github.com:Nanna/usegit.git (push)
```

对比github：



解决方法：
输入以下两个命令行：
$ git remote set-url origin https://github.com/Alreadgo/u...
$ git push -u origin master
问题解决，最后git bash输出如下

```
Counting objects: 29, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (25/25), done.
Writing objects: 100% (29/29), 2.40 KiB | 615.00 KiB/s, done.
Total 29 (delta 9), reused 0 (delta 0)
remote: Resolving deltas: 100% (9/9), done.
To https://github.com/Alreadgo/usegit.git
 * [new branch]      master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.
```

注意：记得将地址改成自己github上的项目地址！

解决方法https://segmentfault.com/a/1190000015168578

- 从远程库克隆

先在Github创建一个远程库gitskills

克隆本地库

```
$ git clone git@github.com:michaelliao/gitskills.git
```

7. 分支管理

查看分支：`git branch`

创建分支：`git branch <name>`

切换分支：`git checkout <name>`或者`git switch <name>`

创建+切换分支：`git checkout -b <name>`或者`git switch -c <name>`

合并某分支到当前分支：`git merge <name>`

删除分支：`git branch -d <name>`

如果要丢弃一个没有被合并过的分支，可以通过`git branch -D <name>`强行删除

查看远程库信息，使用`git remote -v`；

本地新建的分支如果不推送到远程，对其他人就是不可见的；

从本地推送分支，使用`git push origin branch-name`，如果推送失败，先用`git pull`抓取远程的新提交；

在本地创建和远程分支对应的分支，使用`git checkout -b branch-name origin/branch-name`，本地和远程分支的名称最好一致；

建立本地分支和远程分支的关联，使用`git branch --set-upstream branch-name origin/branch-name`；

从远程抓取分支，使用`git pull`，如果有冲突，要先处理冲突。

ps多人协作模式

多人协作的工作模式通常是这样：

1. 首先，可以试图用`git push origin <branch-name>`推送自己的修改；
2. 如果推送失败，则因为远程分支比你的本地更新，需要先用`git pull`试图合并；
3. 如果合并有冲突，则解决冲突，并在本地提交；
4. 没有冲突或者解决掉冲突后，再用`git push origin <branch-name>`推送就能成功！

如果`git pull`提示`no tracking information`，则说明本地分支和远程分支的链接关系没有创建，用命令`git branch --set-upstream-to <branch-name> origin/<branch-name>`。

8.标签管理

`git tag <tagname>`用于新建一个标签，默认为`HEAD`，也可以指定一个commit id；

`git tag -a <tagname> -m "blablabla..."`可以指定标签信息；

`git tag`可以查看所有标签。

`git push origin <tagname>`可以推送一个本地标签；

`git push origin --tags`可以推送全部未推送过的本地标签；

`git tag -d <tagname>`可以删除一个本地标签；

`git push origin :refs/tags/<tagname>`可以删除一个远程标签。