Git 常用命令和使用

# Git常用命令

## 提交文件到暂存区

git add filename(文件名称)

git rm filename 删除文件

先手动删除文件，然后使用git rm <file>和git add<file>效果是一样的

## 查看状态

git status

## 查看提交历史日志

git log

git log --pretty=oneline

git reflog(查看所有提交命令，包括回退的命令，主要用来回退到历史版本后又想恢复到新版本)

git log –graph可以看到分支合并图

git log --graph --pretty=oneline --abbrev-commit 查看每次提交的历史，包括分支

## 版本回退

git reset --hard HEAD^(回退上一个版本)

git reset --hard HEAD^^(回退上上一个版本)

git reset --hard HEAD~100(回退前100版本)

git reset --hard 版本号(回退到具体的版本号)

git checkout -- filename(丢弃工作区的修改)

git reset HEAD filename(暂存区的修改撤销掉,重新回到工作区)

## 提交到当前分支

git commit -m ""

## 查看区别

git diff HEAD -- filename(文件名称) 查看工作区和版本库里面最新版本的区别

# git和远程仓库的使用

## 创建ssh key

ssh-keygen -t rsa -C “[youremail@example.com](mailto:youremail@example.com)”

在C:\Users\Administrator\.ssh路径下id\_rsa.pub中存放的是秘钥

## 登陆GitHub

打开“Account settings”，“SSH Keys”页面点“Add SSH Key”，填上任意Title，在Key文本框里粘贴id\_rsa.pub文件的内容

本地仓库关联远程仓库

git remote add origin [git@github.com:michaelliao/learngit.git](mailto:git@github.com:michaelliao/learngit.git)

## 推送远程库

git push –u origin(远程库名字) master(本地分支名称)

由于远程库是空的，我们第一次推送master分支时，加上了-u参数，git不但会把本地的master分支内容推送到到远程新的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 origin master

master分支是主分支，因此需要时刻与远程库同步

dev分支时开发分支，团队所有成员都需要在上面工作，所以也需要和远程库同步

bug分支只用于在本地修复bug，就没必要推送到远程了

feature分支是否推送远程，取决于你是否和你的小伙伴合作在上面开发

## 从远程仓库拉取

git pull

当在分支拉取失败时，需要设置本地dev和远程dev的连接

git branch --set-upstream branch-name origin/branch-name

### 本地新建的分支推送到远程仓库

git push origin feature/log(本地分支名称): feature/log(远程分支名称)

## 查看远程仓库信息

git remote

git remote –v 显示更详细的信息

## 从远程库克隆

### 登录githup

创建一个新的仓库，名字为gitskills

勾选Initialize this repository with a README，这样github会自动为我们创建一个readme.md文件

### 克隆

git clone [git@github.com:jin654723383/mytest.git](mailto:git@github.com:jin654723383/mytest.git)

## 删除本地仓库和远程仓库的关联

git remote rm origin

# 分支操作

## 创建dev分支

git checkout –b dev

-b表示创建并切换，相当于一下两条命令

git branch dev 创建分支

git checkout dev 切换分支

创建远程分支到本地

git checkout –b dev(本地分支名称) rigin/dev(远程分支名称)

## 查看分支

git branch

git branch 会列出所有分支，当前分支会加上一个 \* 号

## 合并分支

切换到master分支后,将dev分支内容合并到master上

git merge dev

git merge 用于合并指定分支到当前分支

git merge –no-ff –m “merge with no-ff ” dev

--no-ff 表示禁用Fast forward，合并后的历史有分支，能看出来曾经做过合并

Fast forward合并就看不出来曾经做过合并

## 删除分支

git branch –d dev

git branch –D dev 强行删除

## 暂存

git stash

作用:在分支上进行开发的未完成的时候，同时需要在另外一个分支修改bug，这时需要对当前正在开发的分支暂存 执行git stash

Bug修改完成后，再切换到当前分支

查看暂存

git stash list

恢复暂存有两种方法

一. git stash apply 恢复，但是恢复后stash内容并不会删除,你需要用git stash drop 来删除

二. 用git stash pop 恢复的同时把stash内容也删除了

# 标签

## 创建标签

切换到需要打标签的分支上

git tag v1.0

创建带有说明的标签

git tag –a v1.0 –m “”

查看所有标签 git tag

对历史版本打标签

git tag v1.0 f52c633(commit id)

查看具体标签信息 git show v1.0

## 操作标签

删除标签git tag –d v1.0

删除远程仓库标签 git push origin :refs/tags/v1.0

推送某个标签到远程库 git push origin v1.0

推送全部标签到远程库 git push origin –tags

如果本地标签已经推送到远程，又想删除

先从本地删除 git tag –d v1.0

然后从远程库删除 git push origin :refs/tags/v1.0