一、Git初步学习

1. 安装git

安装后，找到目录下git bash 点击运行，如图安装成功：



1. 设置你的名字和Email地址：

$ git config --global user.name "Your Name"

$ git config --global user.email "email@example.com"

1. 布置一个新仓库

选择一个文件夹，

运行git bash，

输入$ git init , 可以发现当前目录下多了一个.git的目录

1. 给仓库添加并提交一个文件

（提交修改和提交新文件是一样的两步）

添加（到：暂存区） $ git add xxx.txt

提交（到：当前分支） $ git commit -m "wrote a readme file（这是备注）"

1. 时刻查看仓库当前状态

$ git status

1. 查看提交前，修改了什么

$ git diff xxx.txt

1. 关于“工作区”“版本库”



* “版本库”：工作区的一个隐藏目录.git

其中，最重要的是：

“暂存区”stage、

自动创建的第一个分支“master”、

指向“master”的指针“head”

* “工作区”：电脑我们通常编辑的目录文件

1. 版本回退

上版本 $ git reset --hard HEAD^

上上版本 $ git reset --hard HEAD^^

上100个版本 $ git reset --hard HEAD~100

指定版本 $ git reset --hard id

1. 查看日志

查看提交日志（近=>远），可以获取版本id和提交明细：

提交日志详细 $ git log

最近一次提交 $ git log -1

提交日志简化 $ git log --pretty=oneline

分支合并图 $ git log --graph

分支合并图 $ git log --graph --pretty=oneline --abbrev-commit

分支合并表 $ git log --pretty=oneline --abbrev-commit

命令日志 $ git reflog

10、撤销修改

$ git checkout -- xxx.txt 撤销“工作区”修改

若自修改后还没有被放到暂存区，现在，撤销修改就回到和版本库一模一样的状态；

若已经添加到暂存区后，又作了修改，现在，撤销修改就回到添加到暂存区后的状态。

总之，就是让这个文件回到最近一次git commit或git add时的状态。

$ git reset HEAD xxx.txt 撤销“暂存区”修改

该命令既可以回退版本，也可以把暂存区的修改回退到工作区，当我们用HEAD时，表示最新的版本。

11、删除文件

删除 $ git rm xxx.txt

提交 $ git commit -m "remove xxx.txt"

12、配置别名

如，用st表示status $ git config --global alias.st status（加上--global是针对当前用户起作用的，如果不加，那只针对当前的仓库起作用）

<很多人都用co表示checkout，ci表示commit，br表示branch>

二、Git连接远程仓库

1. 注册GitHub账号
2. 本地创建SSH Key：

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

找到id\_rsa（私钥）和id\_rsa.pub（公钥）这两个文件

1. 登录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：

GitHub只要知道了你的公钥，就可以确认只有你自己才能推送。



1. 创建GitHub个人仓库

在右上角找到“Create a new repo”按钮，创建一个新的仓库：



1. 关连本地仓库与远程仓库

把一个已有的本地仓库与之关联，然后，把本地仓库的内容推送到GitHub仓库：

$ git remote add origin git@github.com:账号名/仓库名.git

1. 推送本地分支到远程分支

第一次推送： $ git push -u origin master（推送分支名）

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。

日后推送： $ git push origin master（推送分支名）

如有必要，建立与远程分支关联：

$ git branch --set-upstream-to=远程分支名 本地分支名

抓取远程分支最新版本 $ git pull

解决冲突，提交，再push推送分支名

1. 从远程库拷贝至本地库

从零开发，最好先创建远程库，然后再克隆至本地。

克隆至本地： $ git clone git@github.com:账号名/仓库名.git

1. 查看远程仓库信息

查看： $ git remote

详细查看： $ git remote -v

三、Git使用分支

1、分支创建与合并图示

* HEAD指向当前分支，master指向主分支



* 创建并切换新分支dev，HEAD则指向当前分支dev



* 对当前新分支dev修改并提交，dev指针前移一步



* dev合并到master，需将master指向dev的提交



* 删除dev分支，HEAD指向master分支为当前分支



2、查看当前分支

$ git branch

3、创建、切换、合并、删除分支

创建并切换 $ git checkout -b 分支名 （已过时）

创建并切换 $ git switch -c 分支名

切换分支 $ git switch 分支名

创建分支 $ git branch 分支名

合并分支 $ git merge 分支名（fast forward模式）

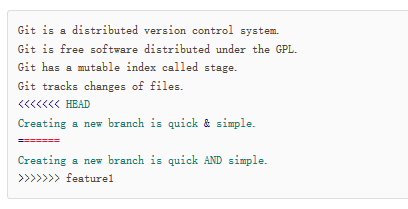
合并分支 $ git merge --no-ff -m "备注" 分支名（普通模式，非fast forward模式）

删除分支 $ git branch -d 分支名

强删未合并分支 $ git branch -D 分支名

4、合并遇到冲突

查看冲突文件，发现：



Git用<<<<<<<，=======，>>>>>>>标记出不同分支的内容

5、BUG分支

修复bug流程：

* 储存当前工作现场副本： $ git status （清空当前工作区）
* 获取master最新版本： $ git checkout master （恢复工作区至master）
* 创建并切换bug分支： $ git switch -c 分支名
* 修复bug文件，提交： $ add、$ commit
* 切换至master分支： $ git switch master
* 合并bug分支： $ git merge --no-ff -m "备注" bug分支名
* 切换至工作分支： $ git switch 分支名
* 还原工作现场副本：

查看所有工作现场 $ git stash list

恢复工作现场{0} $ git stash apply stash@{0} （恢复的stash不被删除）

$ git stash pop （恢复的stash同时删除）

手动删除工作现场{0} $ git stash drop stash@{0}

* 若需要，可应用修改到当前分支：$ git cherry-pick 修改bug提交的id

6、分支标签

查看所有标签： $ git tag

查看标签信息： $ git show 标签名

给当前分支当前版本，打标签：$ git tag 标签名

给当前分支指定版本，打标签：$ git tag 标签名 版本id

打标签+备注： $ git tag -a 标签名 -m "备注文字" 版本id

删除本地标签： $ git tag -d 标签名

推送单标签至远程： $ git push origin 标签名

一次性推送全部： $ git push origin --tags

删除标签：

先删除本地标签 ……

再删除远程标签 $ git push origin :refs/tags/标签名