**第一部分git核心与使用**

1、github是一个分布式版本控制软件

每个分支都有一个中央仓库（master），里面存储着所有的代码，即使没有网络，可以用中央仓库进行开发

每个分支都相当于是一个中心，没有主次之分，交互的时候采用中央仓库进行交互。

2、对于每一个分支，

每一个分支都是一个中心版本库，在每个本地库的里面，有本地工作目录，暂存区，版本库，代码先经过暂存区

才可以放入版本库

Push

Pull

版本库

版本库

Commit

Add

暂存区

本地项目（工作区）

默认主分支master

Ssh

免秘钥

Ssh

默认主分支master

第一步是用git add把文件添加进去，实际上就是把文件修改添加到暂存区；

第二步是用git commit提交更改，实际上就是把暂存区的所有内容提交到当前分支。

因为我们创建Git版本库时，Git自动为我们创建了唯一一个master分支，所以，现在，git commit就是往master分支上提交更改。

Git pull https://github.com/zengzengxin/git\_test.git从远程分支拉取代码

Git push https://github.com/zengzengxin/git\_test.git将代码提交到远程分支

使用过程，

1. 安装git
2. 配置用户名和邮箱

配置用户名 git config --global user.name “用户名”

如：git config --global user.name “zengzengxin”

配置邮箱 git config --global user.email “邮箱”

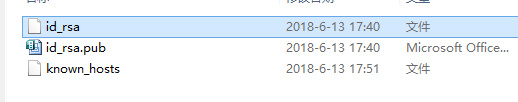
如 git config --global user.email [“1789648780@qq.com”](mailto:\“1789648780@qq.com\”)

1. 搭建远程分支，用github
2. 配置免密钥登陆

1在本地生产ssh，将ssh发送给远程，一人一份，即可

生成密钥

Ssh -keygen -t rsa -c [“1789648780@qq.com”](mailto:\“1789648780@qq.com\”) 一直回车



在github --》setting ——》ssh。。。--》新建秘钥，名字任意，将pub秘钥复制到里面去

检查是否成功连接

Ssh -T [git@github.com](mailto:git@github.com)

得到

Hi zengzengxin! You've successfully authenticated, but GitHub does not provide shell access.（或者ssh里面多一个文件）

则成功

两端通信成功后，进行文件传输

1. 建立本地git项目

1在本地文件夹里打开git barsh，输入git init

2在文件夹里多一个.git文件则成功

2在github上建立一个项目

3将两个项目关联（注意前面的ssh是将两机进行关联，这里是两个项目关联）

git remote add origin https://github.com/zengzengxin/f1test.git

到这里准备工作做完，本机与远程机子关联，本地分支（项目）与远程分支（项目）实现关联。

**第二部分**

进行项目的交互

1. 第一次上传项目（从本地分支到远程分支）
2. git add . //将目录下的所有文件上传到暂存区 （.代表所有文件）
3. git commit -m “第一次上传” //见文件上传到本地库，-m后面的为本次上传的原因 3、git push -u origin master 将文件上传到远程分支

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。

当发现冲突的时候先合并一下代码

可以通过如下命令进行代码合并【注：pull=fetch+merge]

git pull --rebase origin master

然后再用上面的命令进行上传

2、第一次下载项目

git clone [git@github.com:zengzengxin/f1test.git //git](mailto:git@github.com:zengzengxin/f1test.git/git) clone +项目唯一标识符

3本地修改项目后提交到远程

（在工作目录 右键git barsh）

1 git add . //提交到暂存区

2 git commit -m “更新了几行” //提交到本地库

3 git push origin master //提交到远程，比第一次提交少了一个-u

4、将远程分支的东西更新本地

git pull

1. 切换分支与创建分支

A remote：查看现在正在连接着的远程分支

新建一个分支（develop）

Git branch develop

B查看当前所有分支

Git branch

C切换到某一个分支

Git checkout develop

D合并到指定分支上（当我们在develop上做修改并提交到本地分支库（还没有push）的时候master分支并没有修改，merger命令可以使master分支更新到和指定的分支一个点（这里指定的是develop））

Git checkout master //先切换会master

Git merger develop //更新到develop

查看远程库的信息

Git remote -v

推送到远程（将本地的dev分支上的东西推送到远程）

Git push origin dev

但是，并不是一定要把本地分支往远程推送，那么，哪些分支需要推送，哪些不需要呢？

master分支是主分支，因此要时刻与远程同步；

dev分支是开发分支，团队所有成员都需要在上面工作，所以也需要与远程同步

从远程抓取分支（这时候我们只能够从远程的master上克隆）

当提交有冲突的时候，先从远程把最新的代码pull下来，合并一下冲突，然后提交

从dev上抓取代码

我们在使用 git clone  xxx.git 下载代码的时候，获取到的只是 master上的代码

　　假入有个 dev 分支我们想获取上面的代码怎么办！

　　#下载dev分支上的代码并切换到dev分支

git checkout -b dev origin/dev

克隆远程非master分支上的代码

Git clone -b develop +地址

查看冲突部分

Git diff

取回特定分支的更新

Git fetch

**6多人合作开发**

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。

1. **在myeclipse下操作git 使用egit**

A 在window--》preference-->team-->git-->configuration 配置邮箱和用户名

B general -network -ssh2 选中生成的ssh

1、第一次发布项目

a在项目上右键 share project 选择github

b将项目天际到暂存区 右键--》team--》add to index

c提交本地分支 commit

D 将项目推送到远程 右键--team--》reomte --》push 依次填就可以了

2、修改后再次提交

A team -->add to index

B team -->commit

C team-->push

提交的时候

Commit 和commit and push 的区别

Commit :

会将整个项目先提交到本地分支，然后整个文件提交过去

Commit and push ：

可以单独提交某一个被你修改的文件，然后只将这个被修改的文件进行提交

3、第一次下载项目

Import--》git--》clone--》输入https/ssh，一路下一步即可

4、将远程的分支更新到本地

对要更新的项目，右键--》pull--》ok

Git log 查看commit里面保存了哪些版本

git reset 版本回退，退回到上commit一个版本

git reflog 可以查看所有的历史版本编号

git reset --hard 1094a 根据历史版本编号回退到相应的版本

创建分支的原理

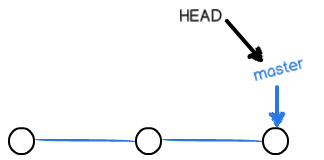
1创建一个dev指针，指向的master同一个点

2将工作指针指向dev

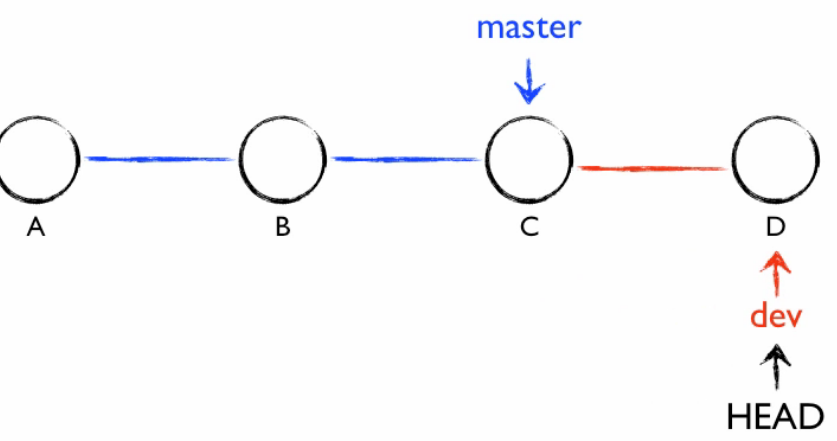
合并dev和master的原理

就是将master指针移动到dev就可以

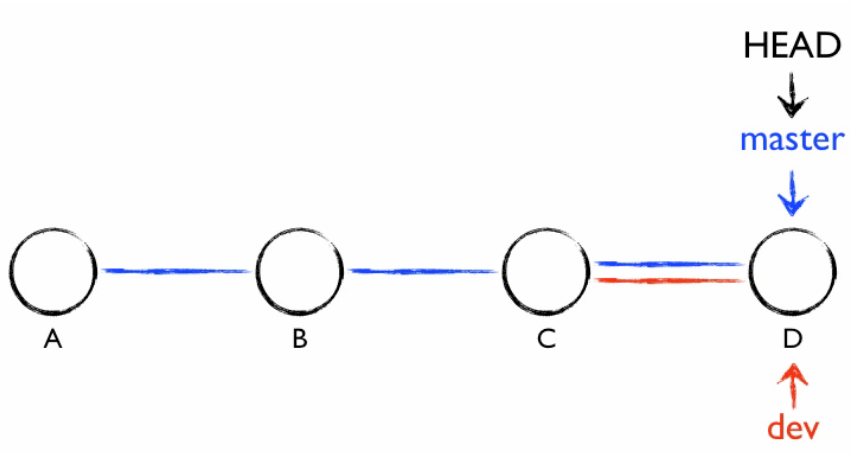
状态一直有master



状态2创建dev后



状态3合并dev和master



创建新的分支并移动到新的分支

Git checkout -b dev

切换分支则用

Git checkout master

查看当前分支

Git branch

在master合并dev分支，然后dev分支会小时，相当于将dev指正的值给了master指针

git merge dev

删除分支：git branch -d <name>

首先，可以试图用git push origin <branch-name>推送自己的修改；

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；

如果合并有冲突，则解决冲突，并在本地提交；

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>。

这就是多人协作的工作模式，一旦熟悉了，就非常简单。

Git常用命令

Git add . //提交全部文件

Git add aaa.text //提交一个文件

反向操作

git reset HEAD XXX.html//撤回一个已经add的文件

Git commit -m “message”//commit全部add的文件提交

反向操作

Git rm -r --cached 111.txt //移除一个已经commit的文件

Git reset --hard head^ //回退整个commit

Git push //提交到远程仓库

反向

Git reset --hard HEAD^ 然后 git push -f 也就是先本地回退，然后强推到远程覆盖

也可以本地修改掉错误的东西，然后再用本地去覆盖远程，为了确保不冲突，可以先pull在push

Git stash //暂存

Git unstash //释放暂存