# git介绍

## SVN和Git的区别

SVN：**是集中式版本控制系统**，版本存放在中央服务器。工作时，从中央服务器取得最新版本，完成工作上交最近版本。必须联网，只在服务器备份，个人电脑上没有版本；

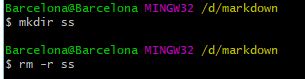
GIT：**分布式版本控制系统**，没有中央仓库，每个人的电脑都是一个完整的版本库，工作时不需要联网，因为版本都在自己电脑上。**协同的方法**：自己的电脑上修改了文件，其他人也修改了该文件，那么只需要互相传给对方就可以知道所有人的修改信息。

# linux命令

* cd： 进入文件夹
* cd ..： 返回上一级
* ls ：列出文件夹下所有文件，ls（II）列出的内容更详细，蓝色是目录，绿色是程序，白色是文件

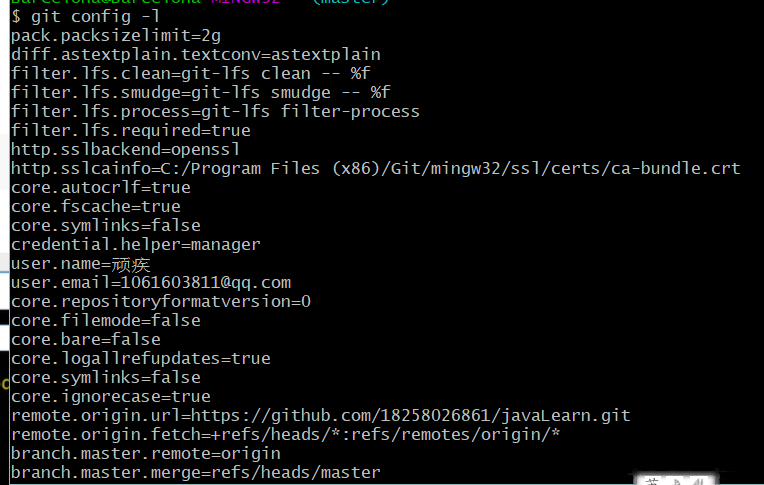
1587383278774

1587383278774

* touch 文件名 ：创建文件
* rm : 移除文件
* mkdir ： 创建一个目录
* rm -r ：移除一个目录
* 
* 1587384390436
* rm -rf / 切勿尝试，清楚根目录的所有文件
* mv A B:把A移到B里
* clear ：清屏
* reset ： 重新加载终端
* history ： 查看历史命令
* exit ： 退出

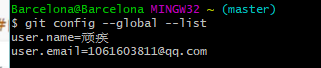
# git配置

#本地账户的所有配置  
git config -l



1587385921324

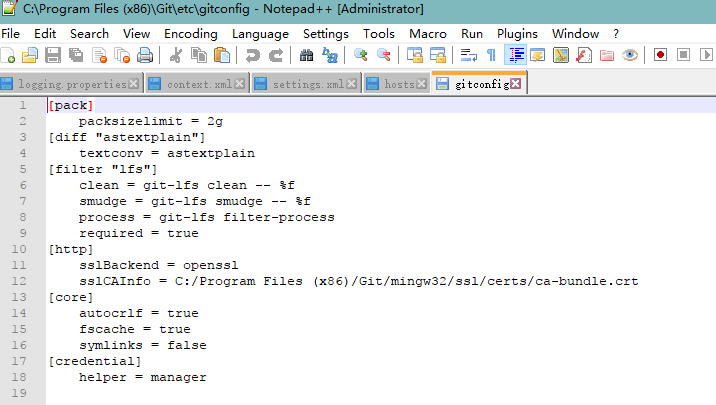
#查看系统config  
git config --system --list  
  
#查看自己的配置，用户名和邮箱是必须配置的  
git config --global --list



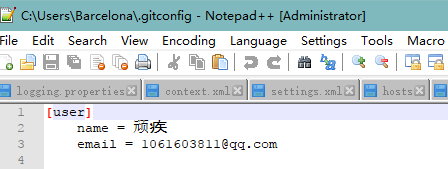
1587385812167

## Git相关配置文件：

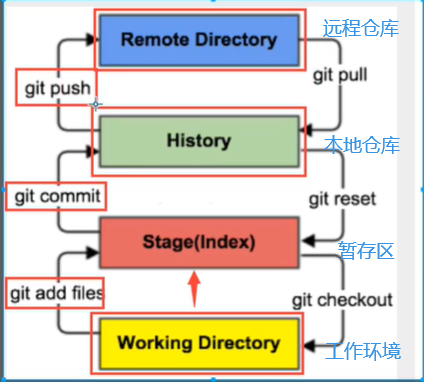
1. **Git安装目录下的gitconfig –system 系统级**

* C:Files (x86)
* 
* 1587388784806

1. **当前登录用户的配置**(==设置用户名和游戏，必要==)：

* C: .gitconfig
* 
* 1587388915527
  + **可以直接在里面写信息**
  + 也可以在git Bash里设置
  + git config --global user.name "顽疾" #名称  
    git config --global user.email 1061603811@qq.com #邮箱

# Git基本理论（核心）



* Woekspace: 工作区，平时存放代码的地方，IDEA项目
* Index/Stage:暂存区，临时存放改动。事实上他是一个文件，保存即将提交到文件列表信息
* Repository:本地仓库，安全存放代码的地方，这里面有所有版本信息。其中HEAD指向最新版本
* Remote:远程仓库，托管代码的服务器

## 创建本地仓库

1. 创建全新仓库，需要用Git管理的根目录下执行

* 在当前目录新建一个Git代码库  
  $ git init
* 1587390826810
* 1587390826810
* 然后多出了一个.git目录，

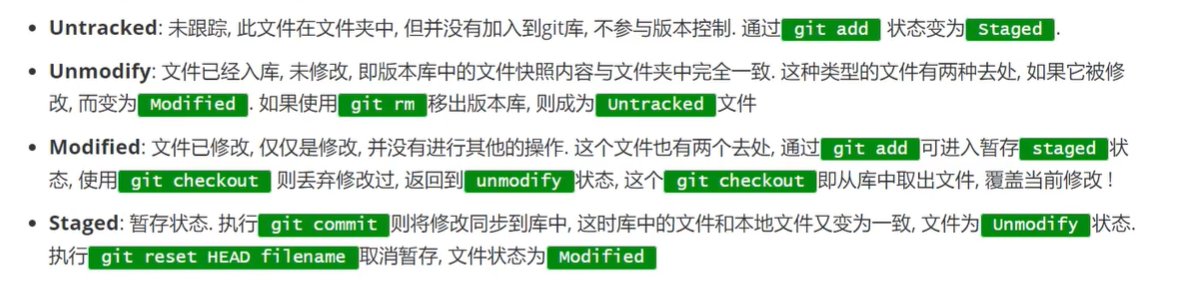
1587390898008

1587390898008

1. 克隆远程仓库，将远程仓库镜像一份到本地

* # 克隆一个项目和整个版本信息  
  $ git clone [url] #clone的网址在github仓库中有

## 文件操作

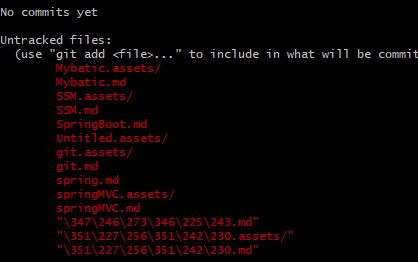


1587391890773

1.创建仓库后**查看仓库文件信息**

# 查看指定文件状态  
$ git status [filename]  
  
#查看所有文件状态  
$ git status

发现都是红的，都没有被放进暂存区

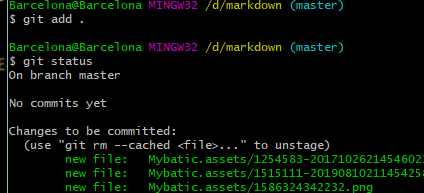


1587391463079

2.将目录下手游文件**放入暂存区**

$ git add .  
# 再查看文件情况  
$ git status

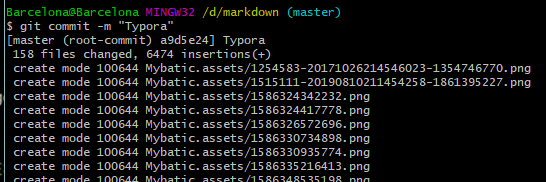
绿色代表在暂存区，但是还没有被**提交**



1587391716565

3.**提交**暂存区的文件到**本地仓库**

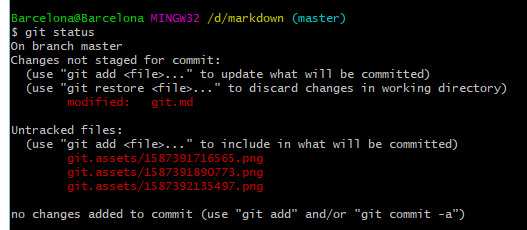
#注意，commit -m一定要加上信息，只写git commit 会跳出一个蓝窗，提示你写信息  
# 这时候ins插入，写完信息之后esc推出，再：wq就行了  
  
$ git commit -m #后面跟的是提交信息



1587392135497

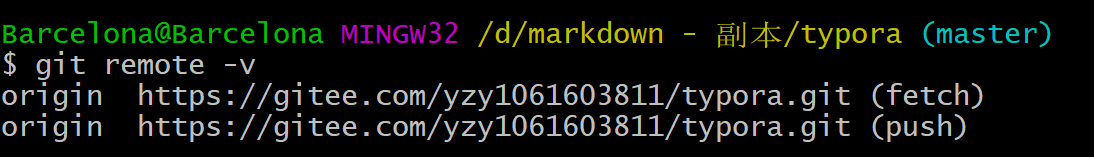
#这时候再查看文件信息  
$ git status

发现绿色的已经提交了，所以不见了（红色是由于我现在一直在编写文档，不停的修改）



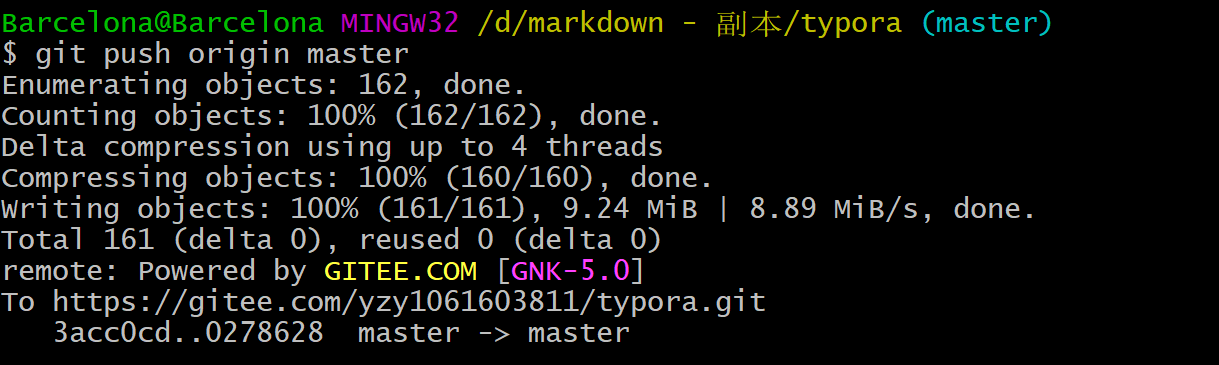
1587392211252

#连接远程仓库  
$ git remote add github https://github.com/18258026861/javaNote.git  
#删除远程仓库  
$ git remote rm origin  
#查看连接的远程仓库  
# 如果连接不上可以克隆远程仓库（根目录需要是需要空文件）一个然后文件复制进去再上传  
$ git remote -v



1587399502227

# origin是别名，有多个远程仓库可以选择别名上传  
git push origin master #origin   
  
# push中出现问题refusing to merge unrelated histories  
  
#这是因为远程仓库已经存在代码记录了，并且那部分代码没有和本地仓库进行关联，我们可以使用如下操作允许pull未关联的远程仓库旧代码：  
  
git pull origin master --allow-unrelated-histories  
  
#强制推送  
git push -u origin master -f



1587399554664

### 忽略文件

有些时候我们不想把某些文件纳入版本控制中，比如数据库文件，临时文件，设计文件等

在主目录下建立“.gitignore”文件，此文件有如下规则：

1. 忽略文件中的空行或以井号（#）开始的行将会被忽略。
2. 可以使用Linux通配符。例如：星号（\*）代表任意多个字符，问号（？）代表一个字符，方括号（[abc]）代表可选字符范围，大括号（{string1,string2,…}）代表可选的字符串等。
3. 如果名称的最前面有一个感叹号（!），表示例外规则，将不被忽略。
4. 如果名称的最前面是一个路径分隔符（/），表示要忽略的文件在此目录下，而子目录中的文件不忽略。
5. 如果名称的最后面是一个路径分隔符（/），表示要忽略的是此目录下该名称的子目录，而非文件（默认文件或目录都忽略）。

#为注释  
\*.txt #忽略所有 .txt结尾的文件,这样的话上传就不会被选中！  
!lib.txt #但lib.txt除外  
/temp #仅忽略项目根目录下的TODO文件,不包括其它目录temp  
build/ #忽略build/目录下的所有文件  
doc/\*.txt #会忽略 doc/notes.txt 但不包括 doc/server/arch.txt

#idea参忽略文件  
# Project exclude paths  
\*.class  
\*.log  
\*.lock  
\*.jar  
\*.war  
\*.ear  
  
/out/  
target/  
tep/  
#idea  
.idea/  
\*.impl

### 生成公钥

**SSH公钥存放处**

C:.ssh

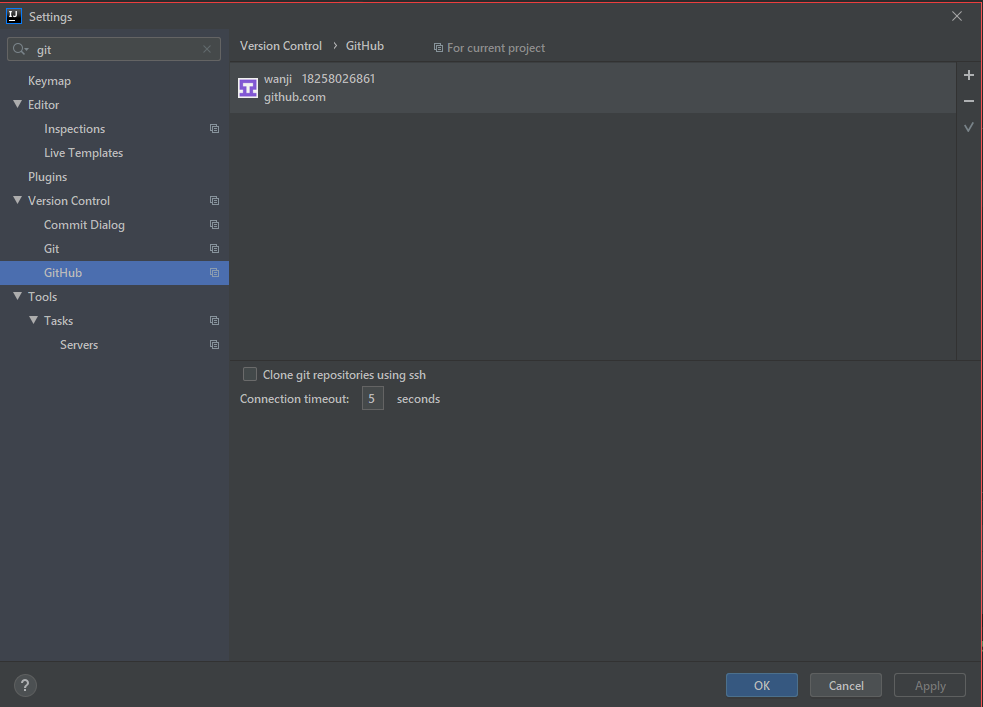
$ ssh-keygen -t rsa

一路回车，会生成文件，将pub文件复制到公钥处即可

# 关于idea提交git的步骤

## 1.连接github

file -> setting -> GitHub

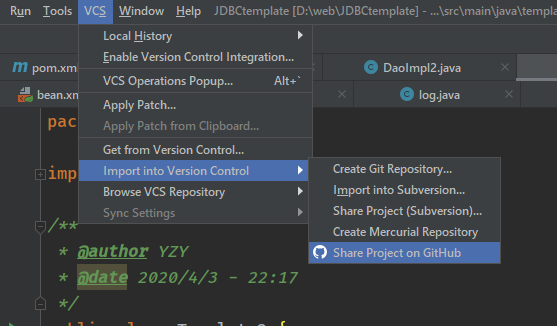


1586405862558

登录github账号

## 2.创建远程仓库

可以使用分享项目到仓库

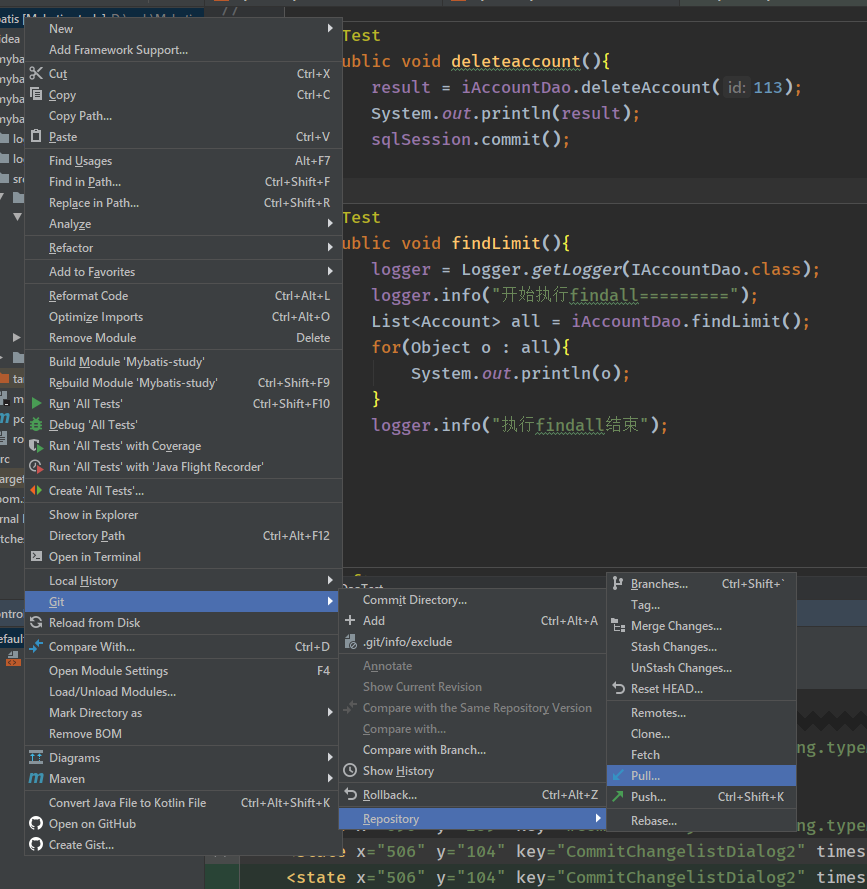


1586406015605

## 3.更新项目

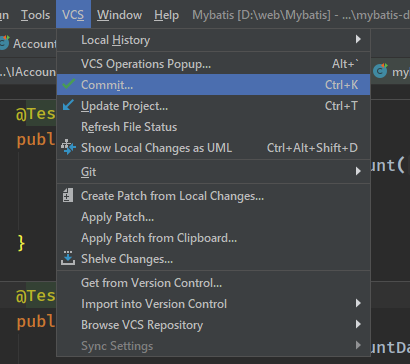
### 1.pull远程仓库

先将github上的项目pull下来（如果对比远程仓库有文件缺失需要这一步）



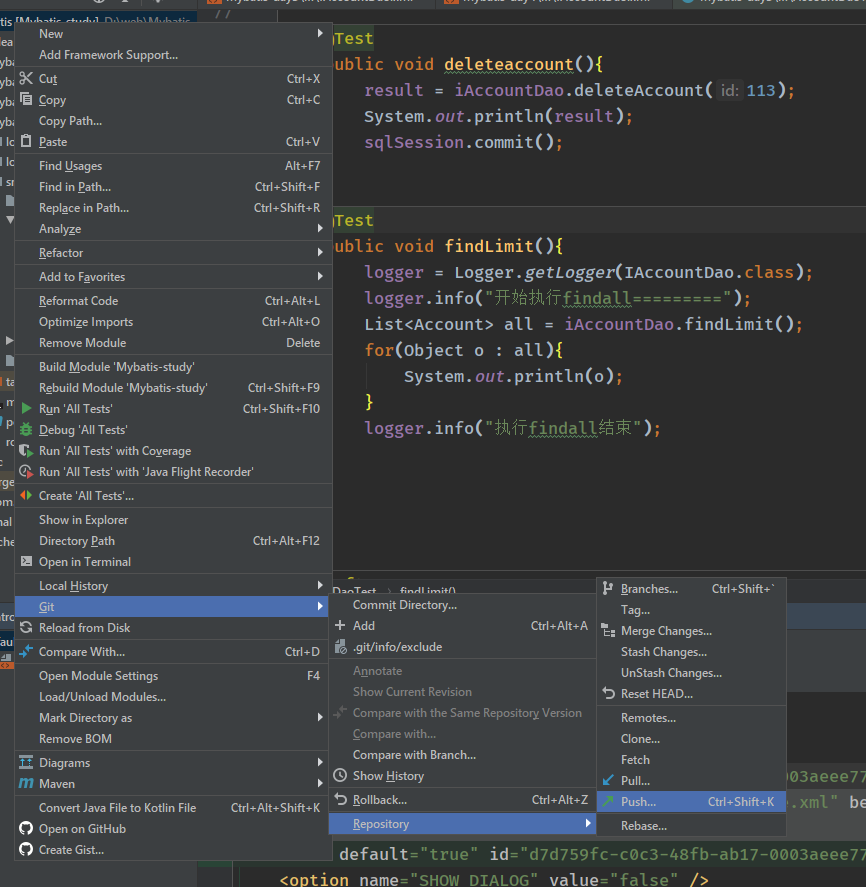
1586406085728

### 2.commit到本地仓库



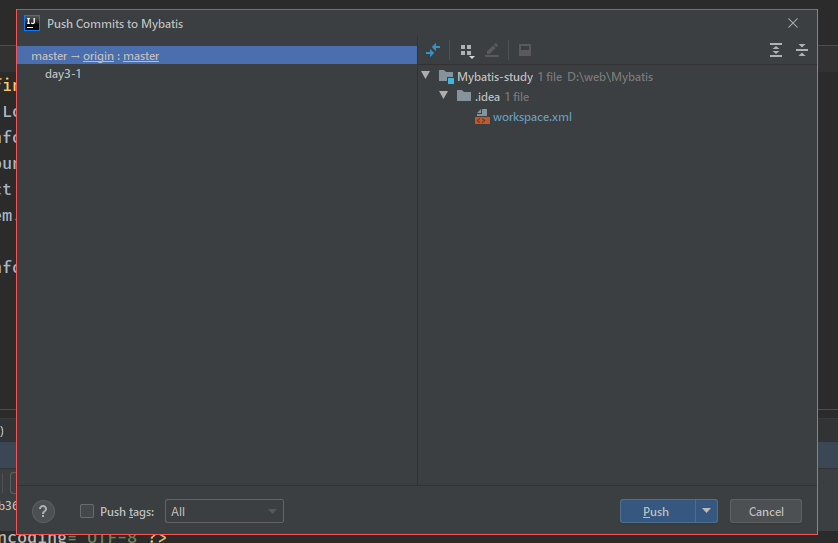
1586406256373

### 3.push



1586406310703

打开之后里面的记录即为本地仓库commit的信息



1586406348111

选择一个push就完成了！

### 也可以用IDEA的命令行(比较方便)

$ git add .  
$ git commit -m ""  
$ git push

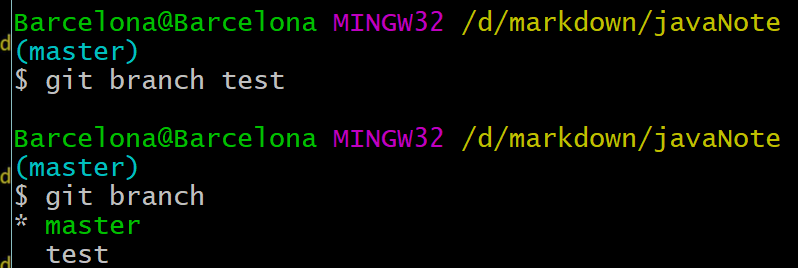
# Git 分支

分支平时互不干扰，但是某个时间他们会合并，就需要处理一些问题

## git 分支指令

# 列出所有本地分支  
git branch  
  
# 列出所有远程分支  
git branch -r  
  
# 新建一个分支，但依然停留在当前分支  
git branch [branch-name]  
  
# 新建一个分支，并切换到该分支  
git checkout -b [branch]  
  
# 合并指定分支到当前分支  
$ git merge [branch]  
  
# 删除分支  
$ git branch -d [branch-name]  
  
# 删除远程分支  
$ git push origin --delete [branch-name]  
$ git branch -dr [remote/branch]

蓝色的代表现在的分支，branch A,创建A分支



1587444614203

如果要在远程仓库创建新分支们可以在本地创建一个分支，然后再在远程创建同名分支。

## 分支冲突

如果多个分支并行执行，就会导致代码冲突，存在多个版本。例如

user和admin两个分支，

A编写user

B编写admin

A编写时需要用到admin.add文件，那么拿过来用的时候**修改**了这个文件，那么user分支里的admin.add文件就修改了。B可以选择是否将这个修改的文件**合并**到admin分支.

## master主分支

master主分支应该非常稳定，用来发布新版本，一般情况下不允许在上面工作，工作一般情况下在新建的dev分支上工作，工作完后，比如上要发布，或者说dev分支代码稳定后可以合并到主分支master上来。