**Git及GitHub详细操作文档**

**目录**

[一、 GitHub团队管理基本配置 2](#_Toc508912950)

[1.1 申请GitHub账号 2](#_Toc508912951)

[1.2 创建新的Organizations（组织） 2](#_Toc508912952)

[1.3 在Organizations中创建Team（团队）并添加成员 3](#_Toc508912953)

[1.4 在Organizations中创建Repositories（仓库）并设置分支 5](#_Toc508912954)

[1.5 在团队仓库项目中添加Team并设置权限 6](#_Toc508912955)

[二、 Git的安装与配置 7](#_Toc508912956)

[2.1 Git的安装 7](#_Toc508912957)

[2.2 配置用户名和邮箱 9](#_Toc508912958)

[三、 使用GitHub及Git 10](#_Toc508912959)

[3.1 创建ssh密钥并在GitHub中配置 10](#_Toc508912960)

[3.2 将团队仓库Fork到私人仓库 11](#_Toc508912961)

[3.3 使用Git将私人仓库项目clone到本地仓库 12](#_Toc508912962)

[3.4 从master分支切换到dev分支 13](#_Toc508912963)

[3.5 将新增的文件add到本地索引表中 15](#_Toc508912964)

[3.6 将本地索引表中新增的文件commit到本地仓库中 16](#_Toc508912965)

[3.7 将本地仓库push到GitHub的私人仓库中 17](#_Toc508912966)

[3.8 将私人仓库请求与团队仓库合并 18](#_Toc508912967)

[3.9 管理员对合并申请进行审核 19](#_Toc508912968)

[四、 使用GitHub及Intellij进行团队开发 20](#_Toc508912969)

[4.1 将私人仓库项目clone到本地仓库 20](#_Toc508912970)

[4.2 在Intellij中配置GitHub及Git 24](#_Toc508912971)

[4.3 master分支切换到dev分支 25](#_Toc508912972)

[4.4 将新增的文件add到本地索引表中 26](#_Toc508912973)

[4.5 将本地索引表中新增的文件commit到本地仓库中 28](#_Toc508912974)

[4.6 将本地仓库push到GitHub的私人仓库中 31](#_Toc508912975)

[五、 使用GitHub及MyEclipse进行团队开发 32](#_Toc508912976)

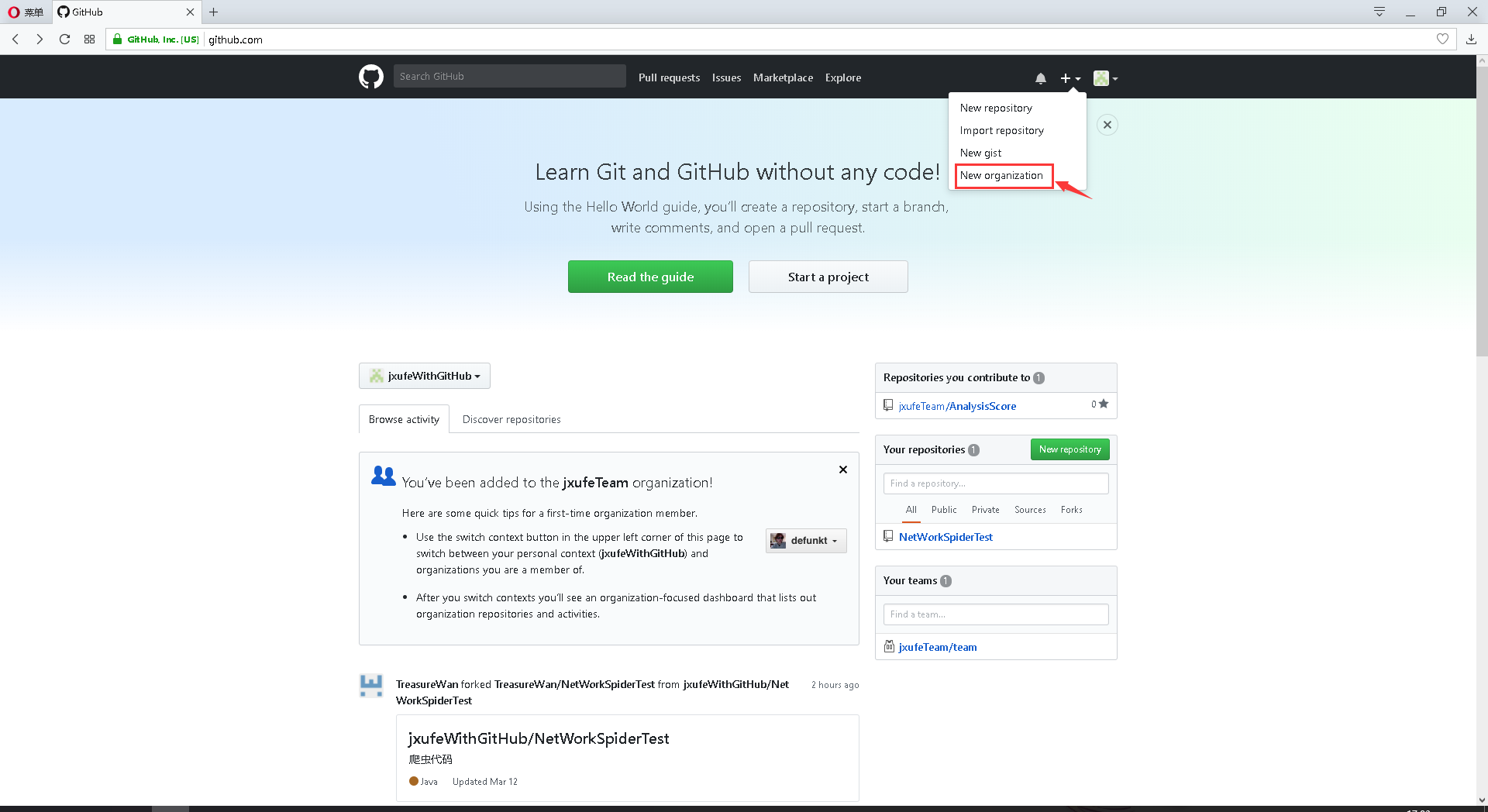
# GitHub团队管理基本配置

## 申请GitHub账号

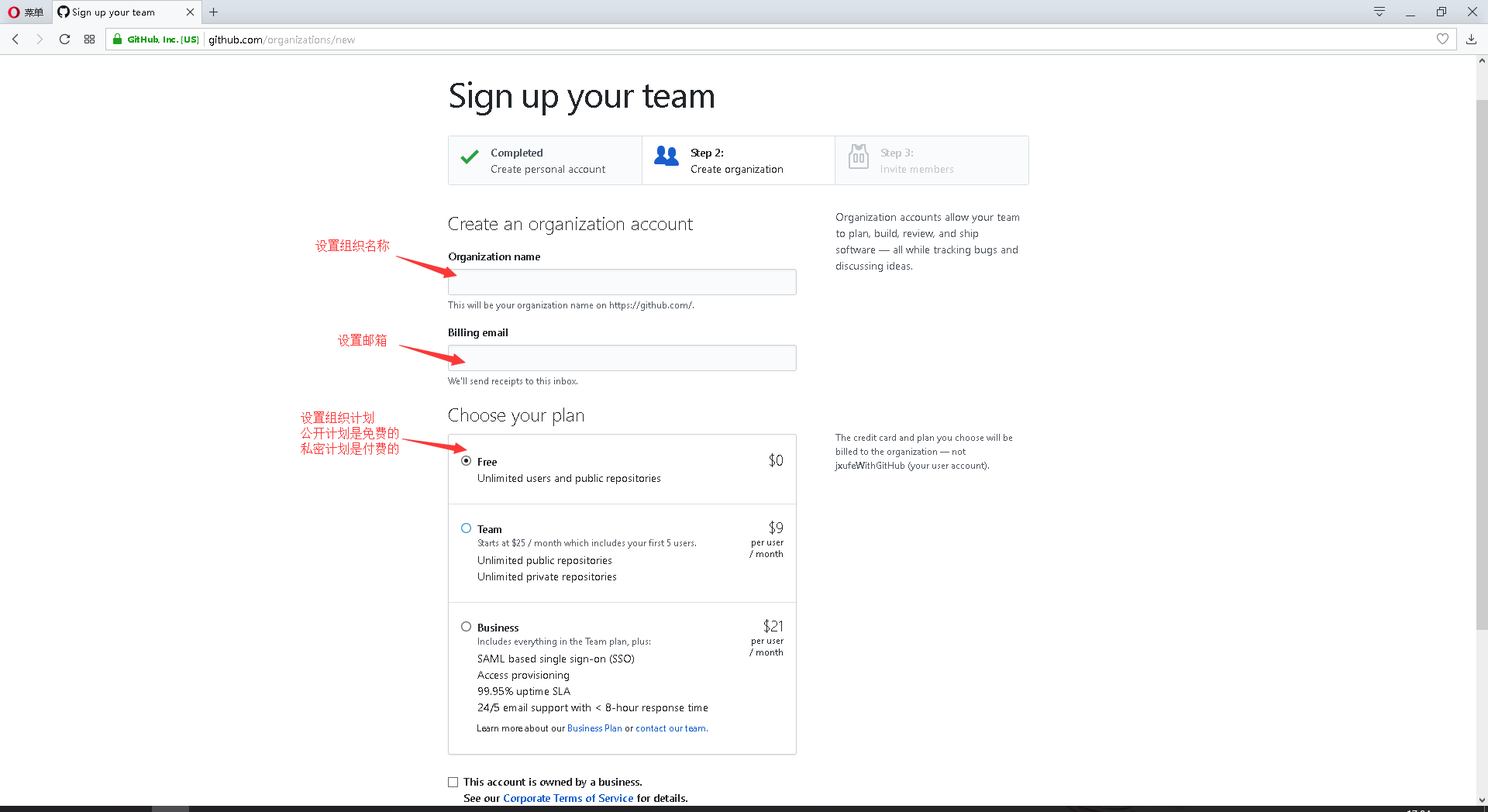
这一步较为简单，不予以说明

## 创建新的Organizations（组织）

在首页中点击下图中的红框处，进入创建Organizations界面

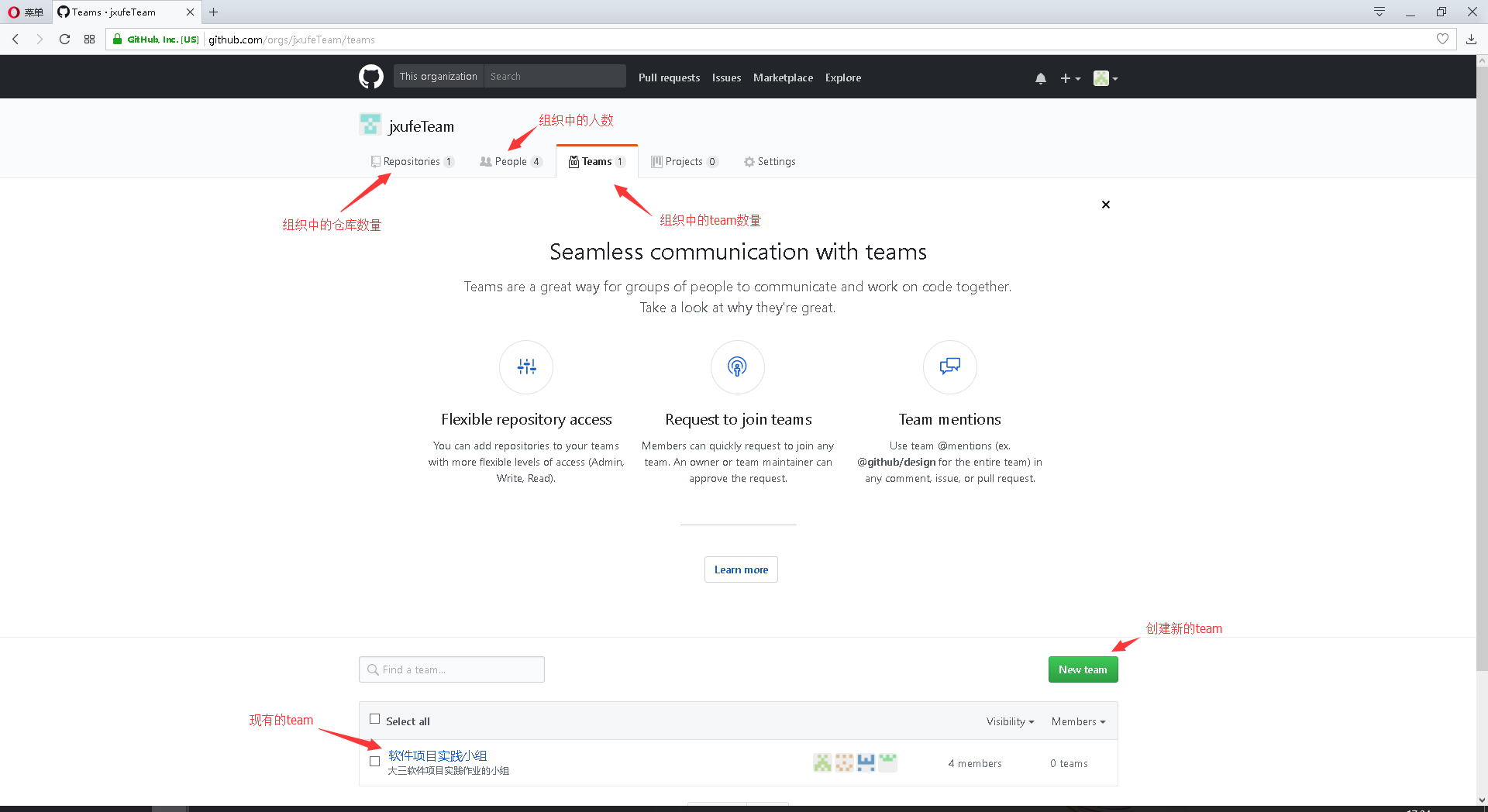


下图为创建Organizations步骤，Invite members步骤可以跳过

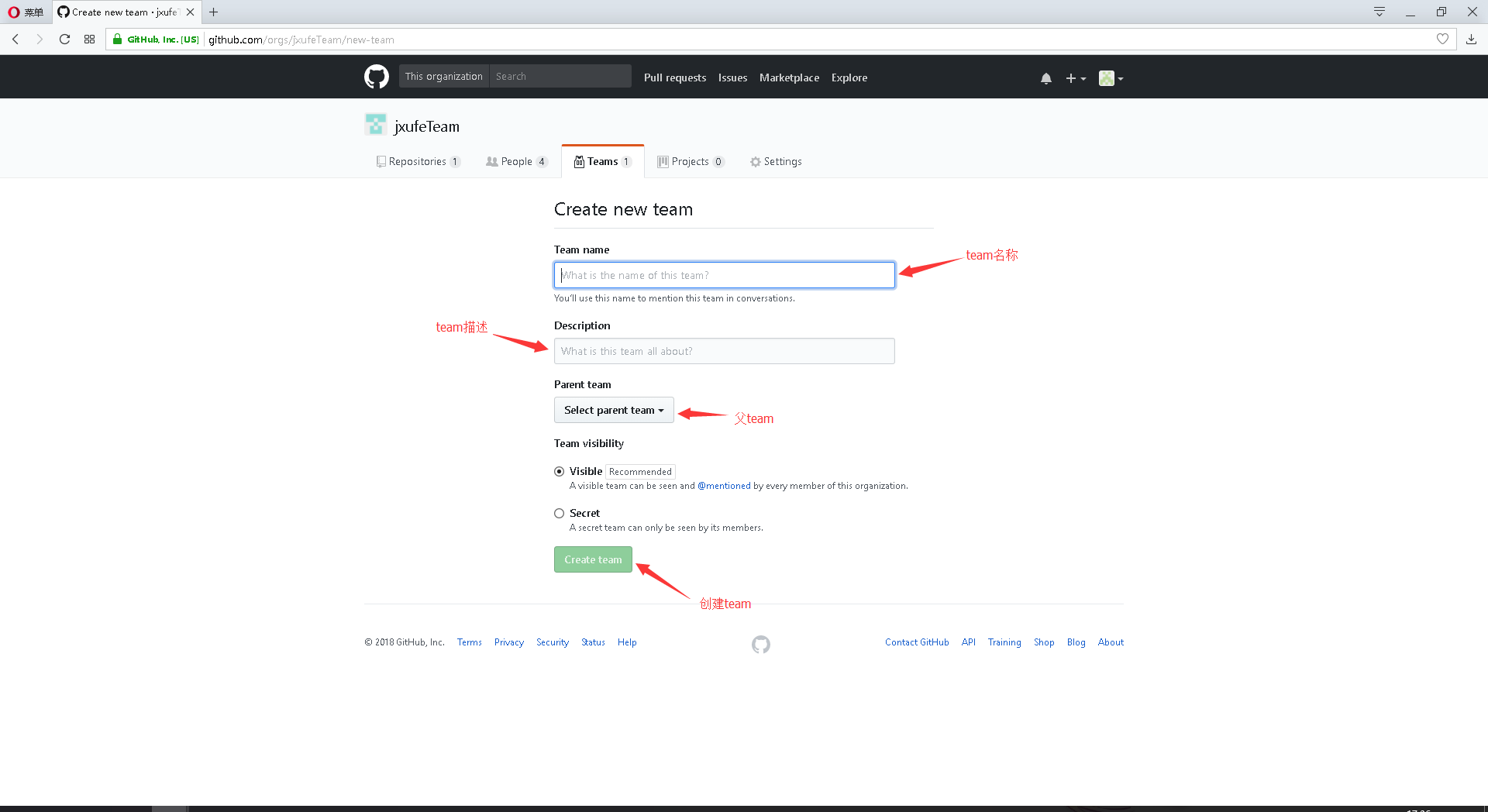


## 在Organizations中创建Team（团队）并添加成员

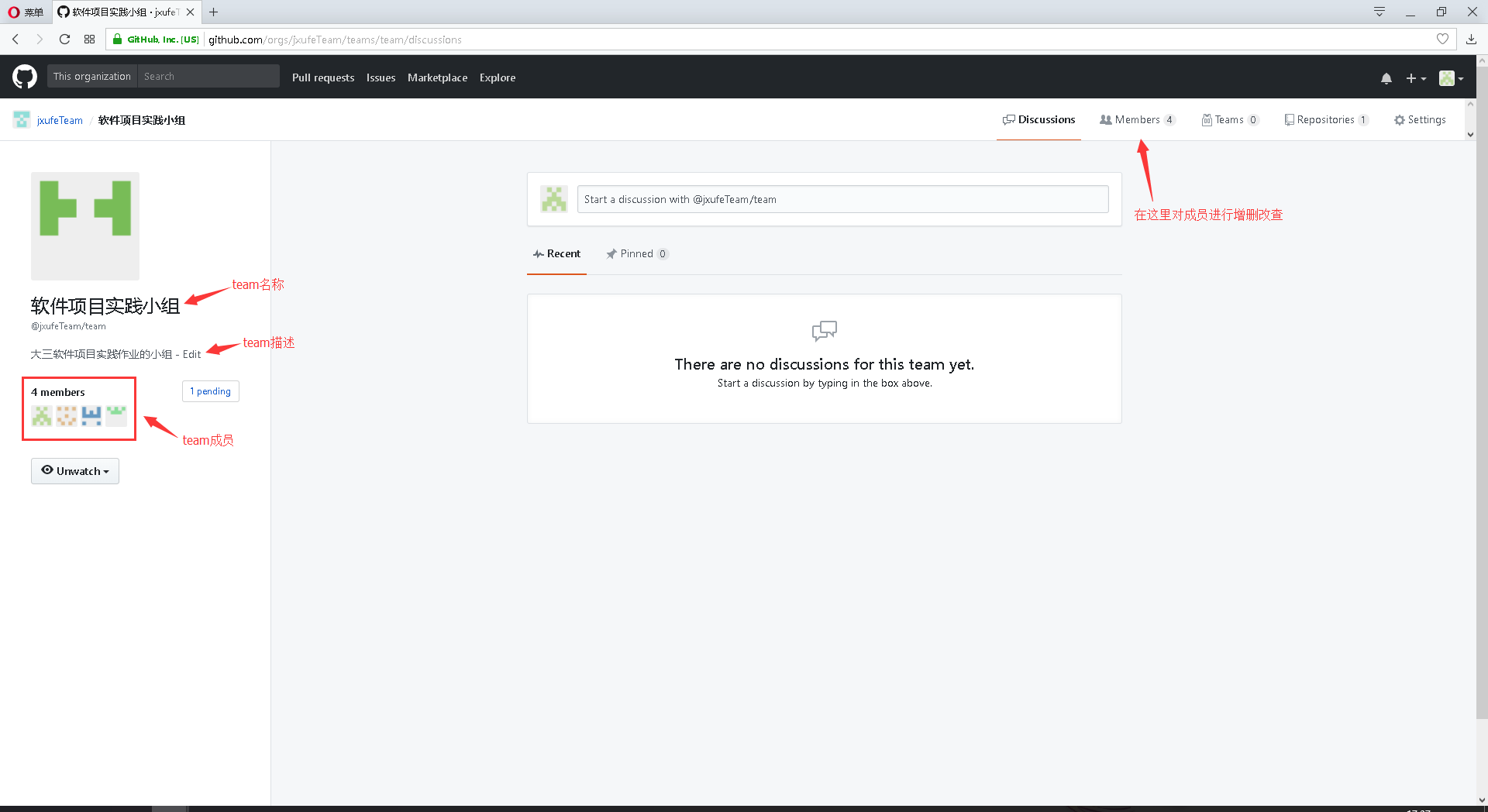
在Organizations界面点击Team => New Team，进入创建Team界面，如下图



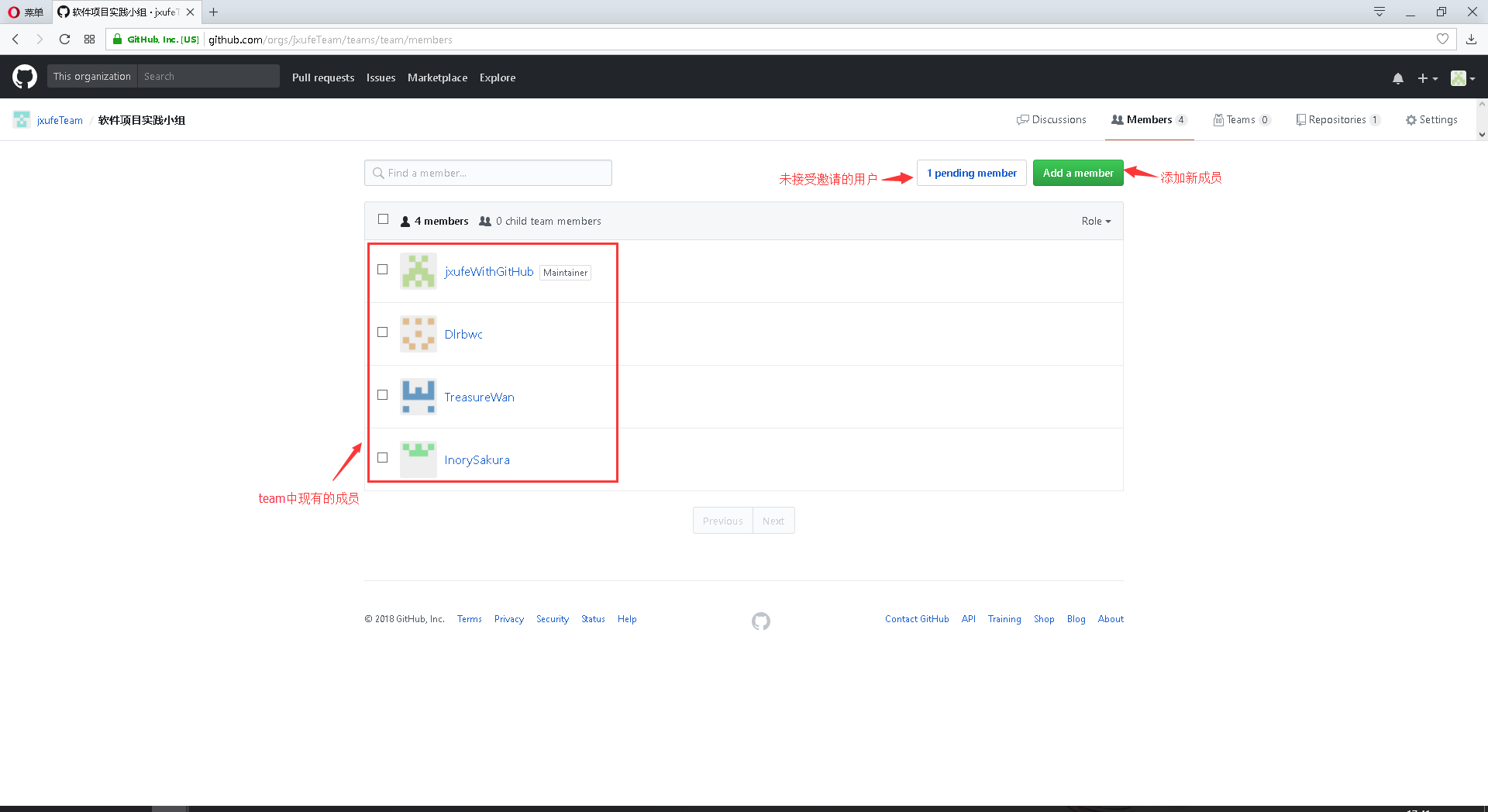
下图为创建Team步骤，填写好必要信息后点击Create team按钮即可创建



创建完成后会自动进入Team界面（如下图），点击下图中的Members，进入添加成员界面

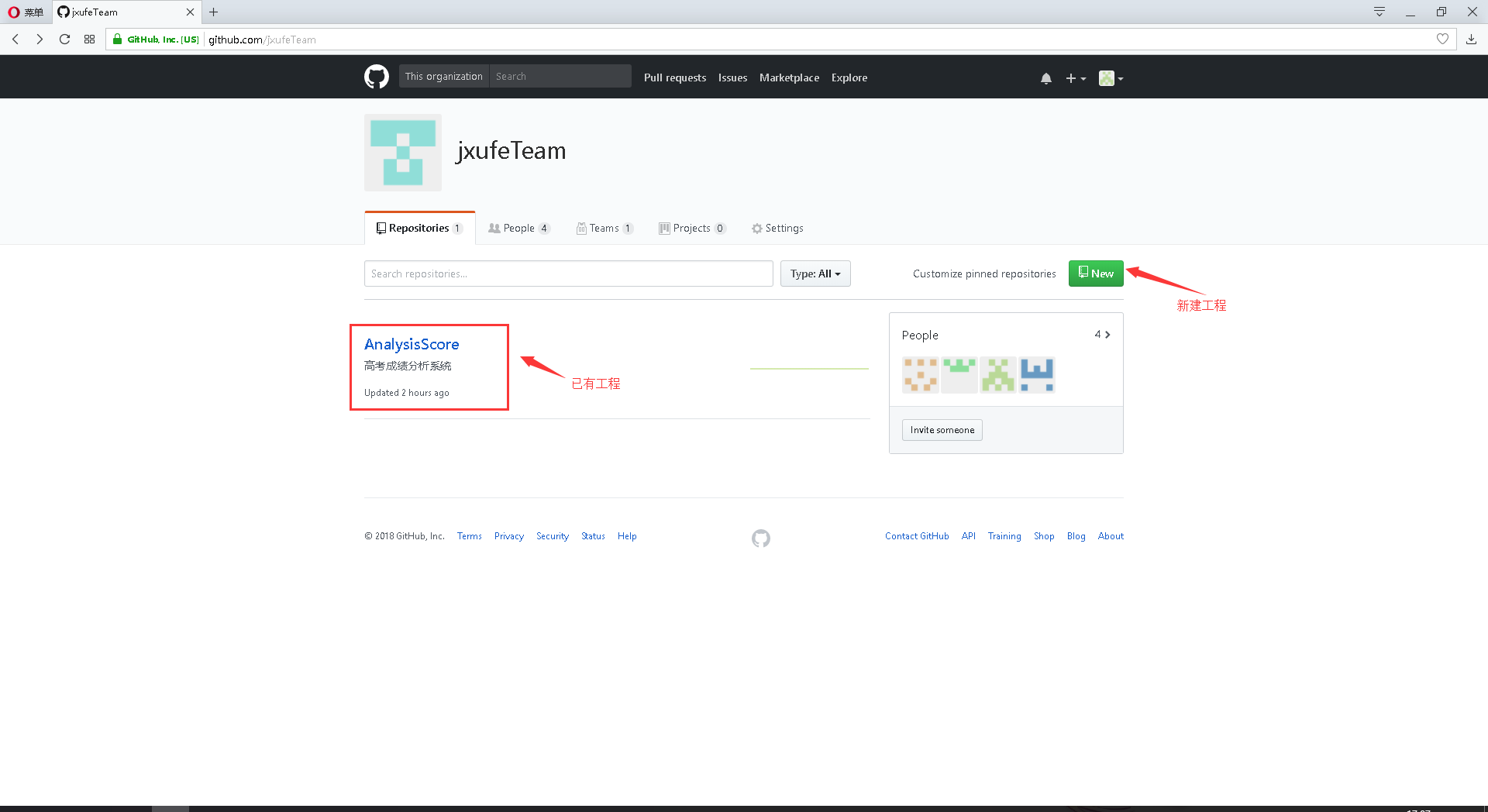


在添加成员界面（如下图）中点击Add a member，输入用户名，即可查找到用户，点击后即可添加

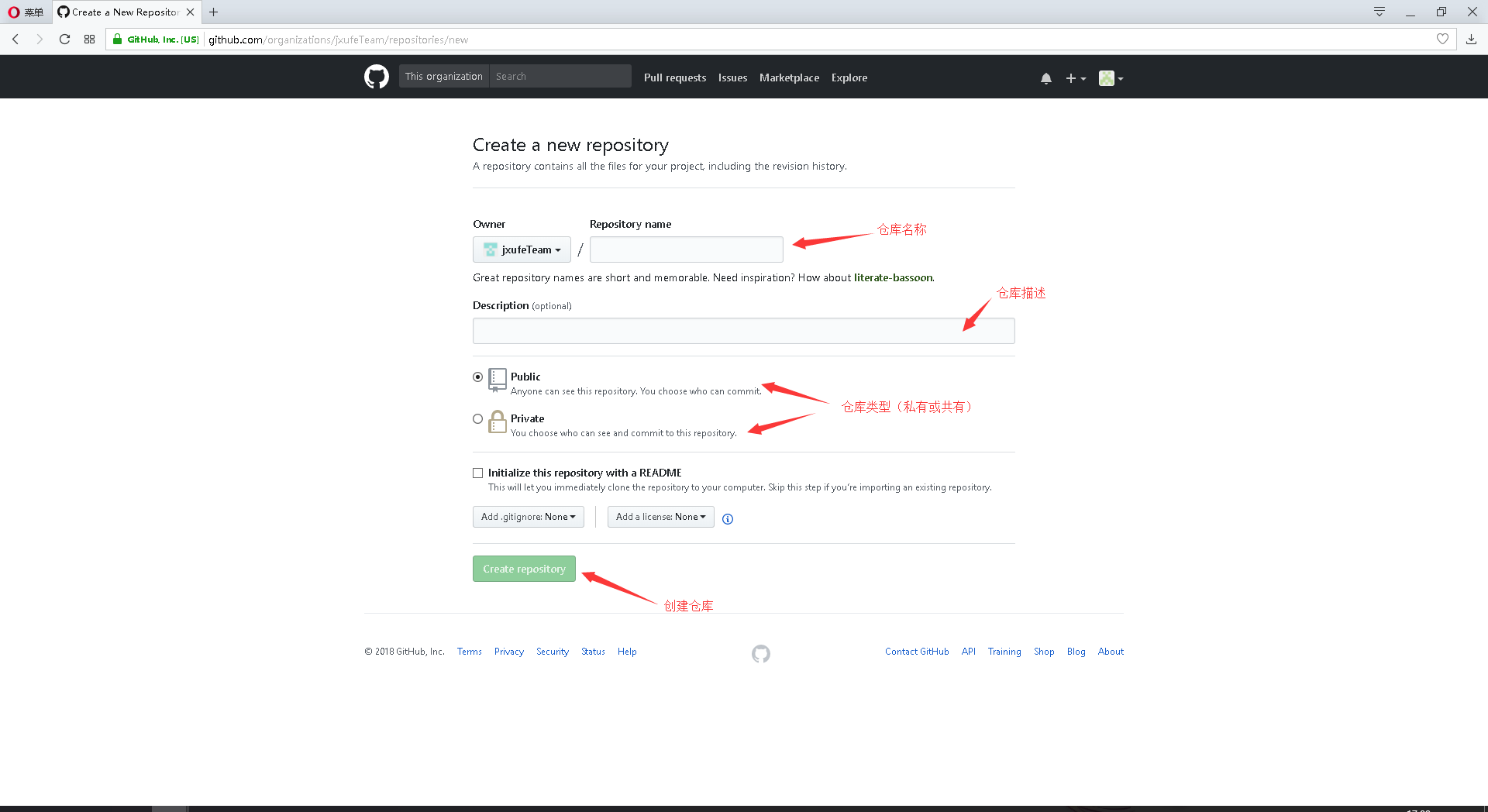


## 在Organizations中创建Repositories（仓库）并设置分支

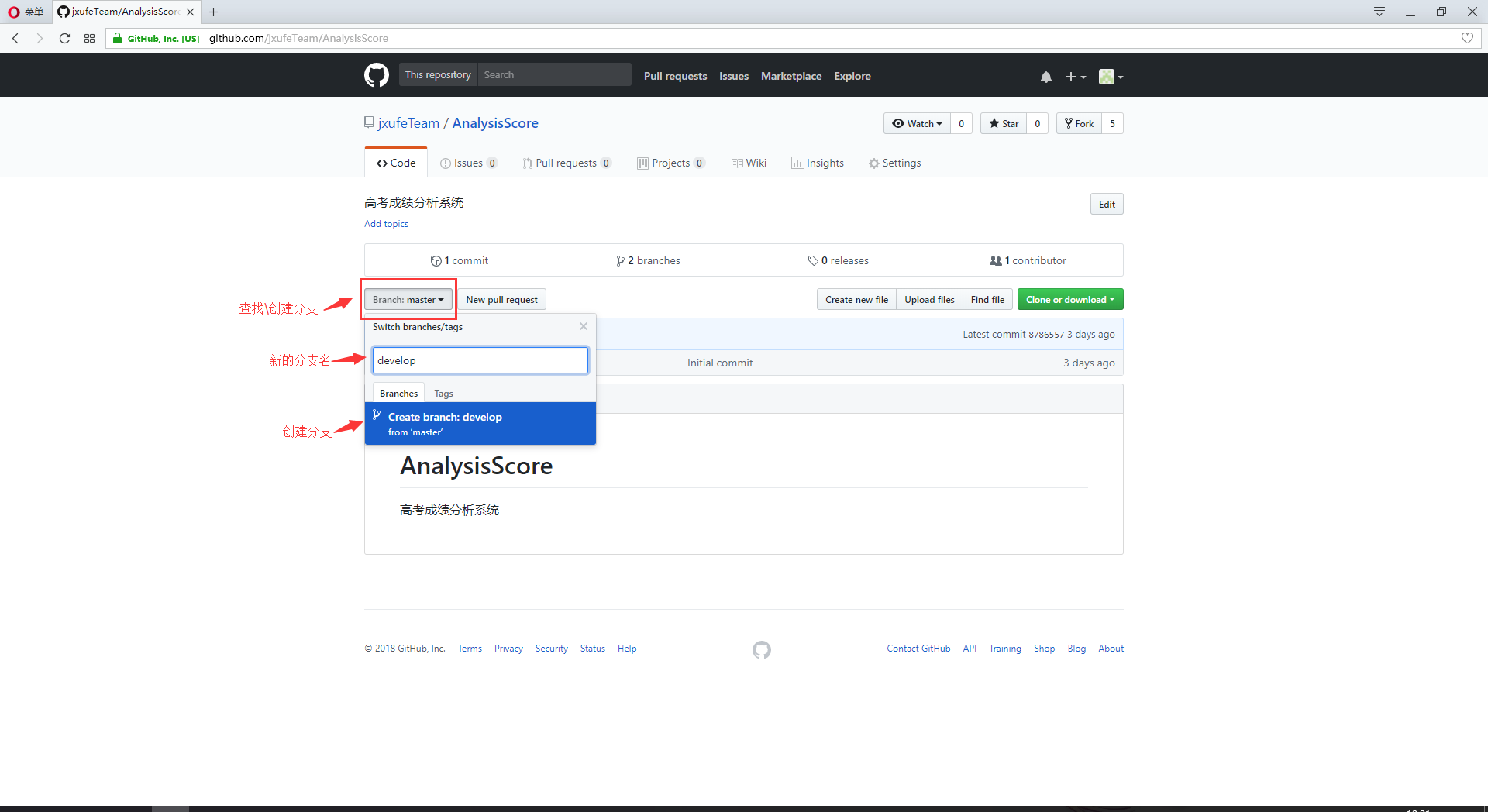
在Organizations界面点击Repositories=> New，进入创建Repositories界面，如下图



下图为创建Repositories的步骤，填写好必要信息后点击Create repositories按钮即可创建



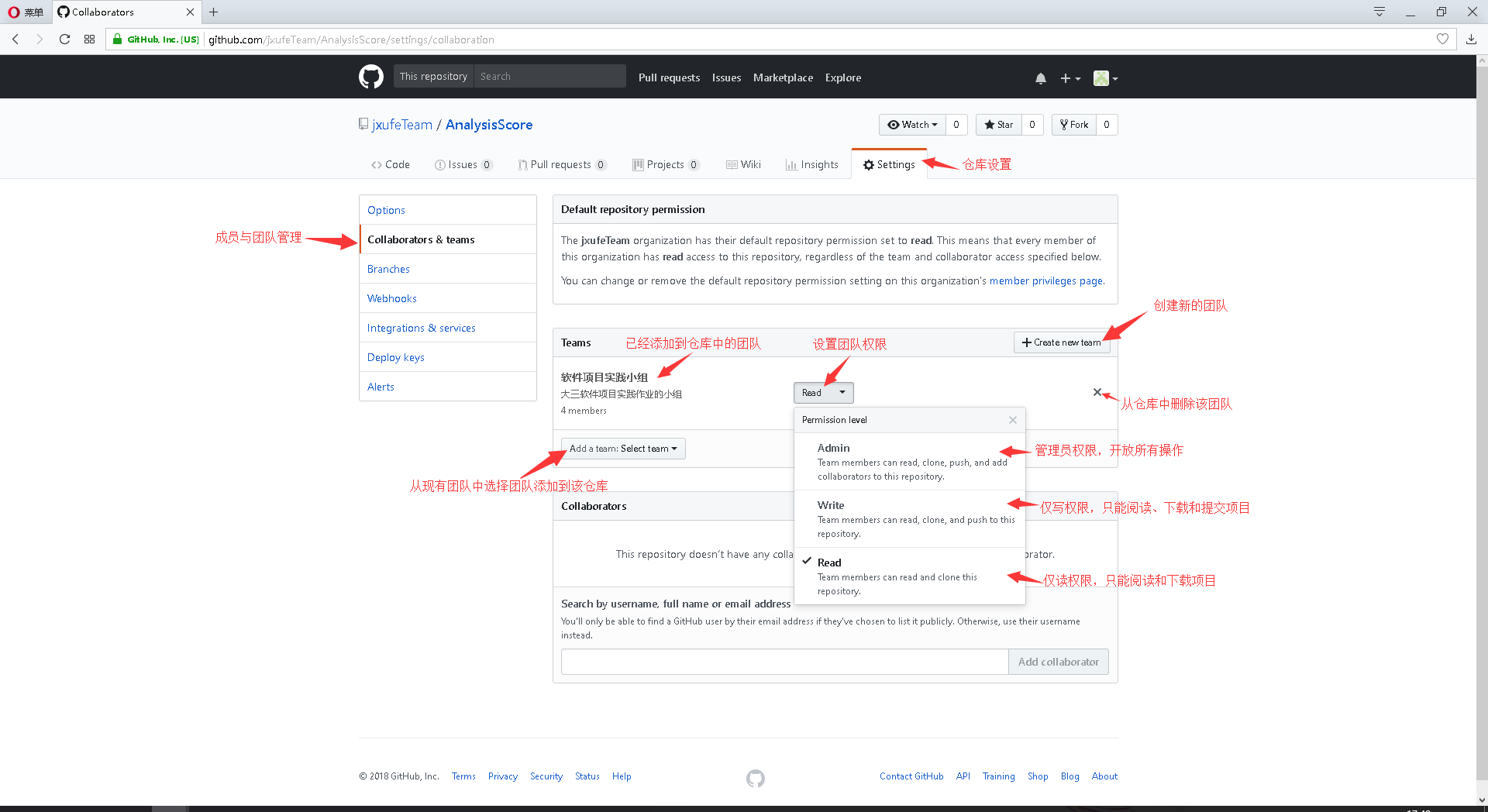
创建完成后会自动进入Repositories界面（如下图），点击下图中的框选的Branch，进入查找\添加分支界面，输入需要创建的分支名，点击Create branch后即可创建分支



这里创建develop（开发）分支的意义在于，我们开发项目时基于develop分支开发并提交项目，此时的项目时处于开发中的项目（不稳定项目），其中可能会存在一些bug。待我们将这个develop分支的项目进行测试消除bug后，在将develop分支合并到master（主）分支上，此时的项目已经时稳定项目了。这样做便于对项目进行版本控制和管理。

## 在团队仓库项目中添加Team并设置权限

在Repositories界面中点击Settings，进入设置界面（如下图），选择左侧的Collaborators & teams，在右侧中的Team里点击Add a team，选择Team、设置好权限并添加到仓库中

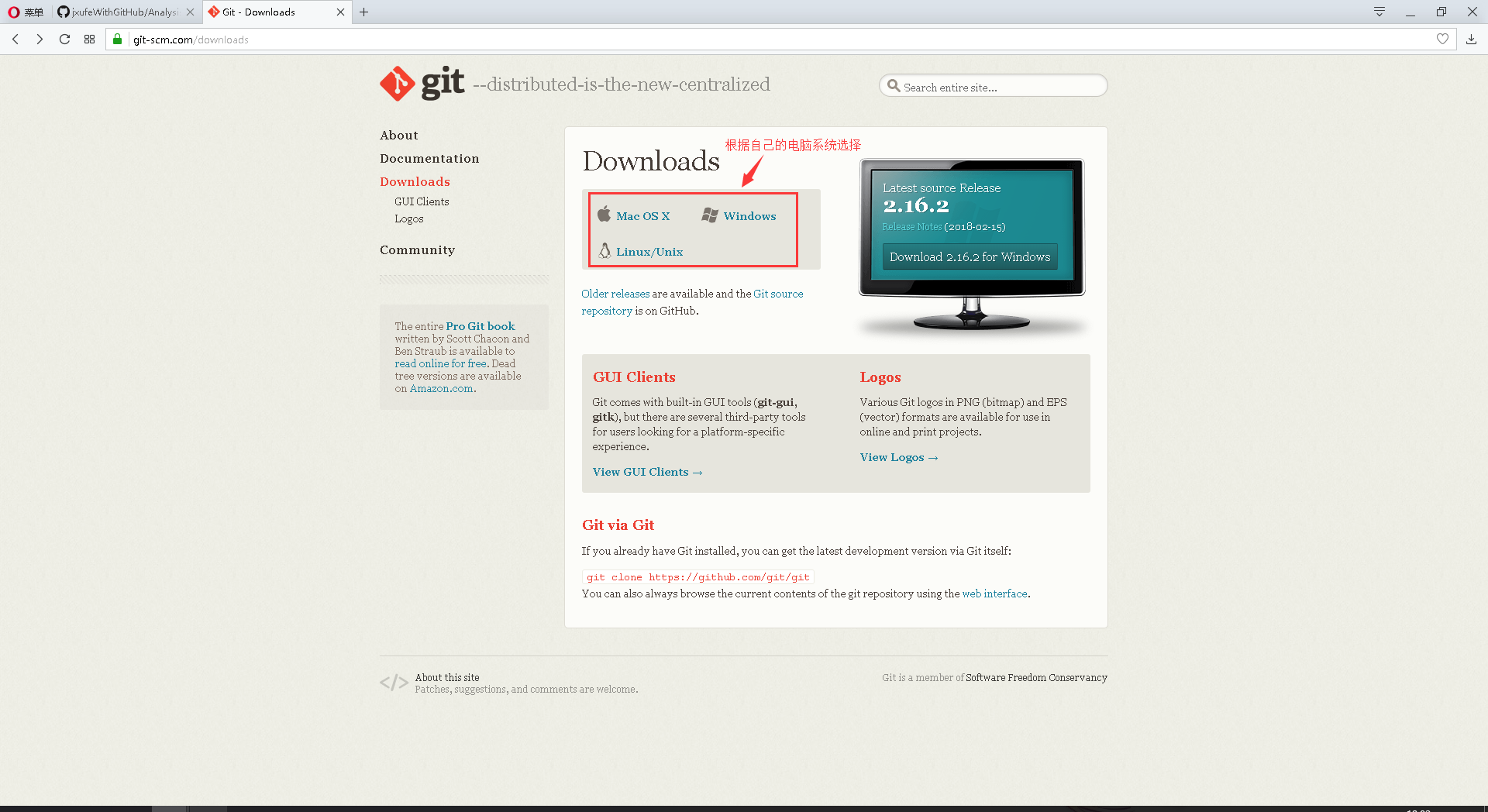


# Git的安装与配置

## 2.1 Git的安装

下载地址：<https://git-scm.com>







Git安装步骤

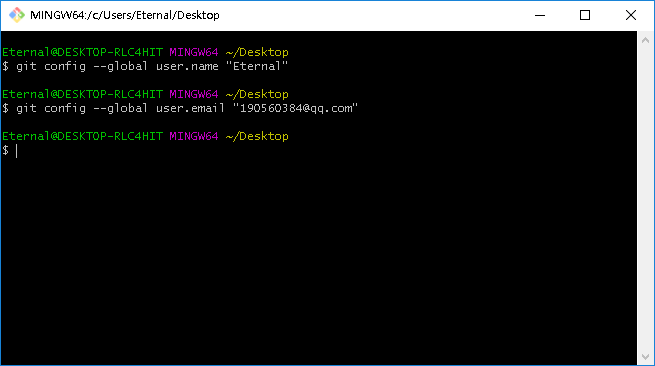


## 2.2 配置用户名和邮箱

分别输入指令：

git config --global user.name “你的GitHub的用户名”

git config --global user.email “你的GitHub的邮箱”

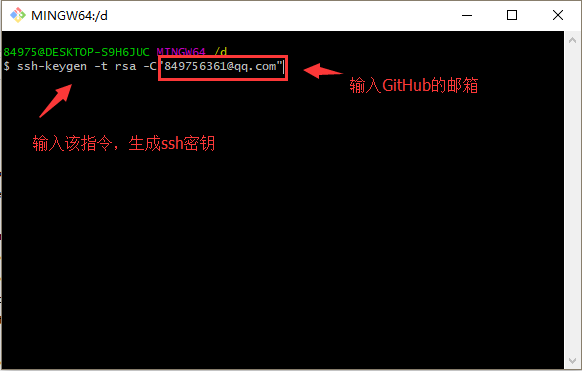


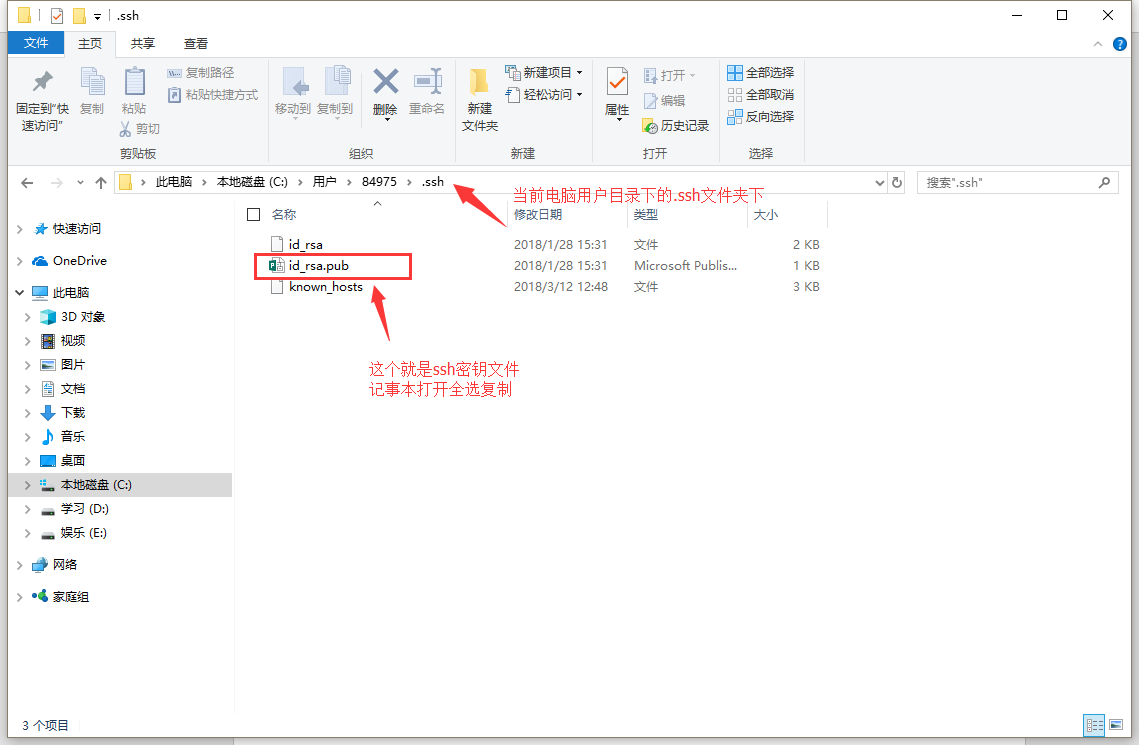
# 使用GitHub及Git

## 3.1 创建ssh密钥并在GitHub中配置

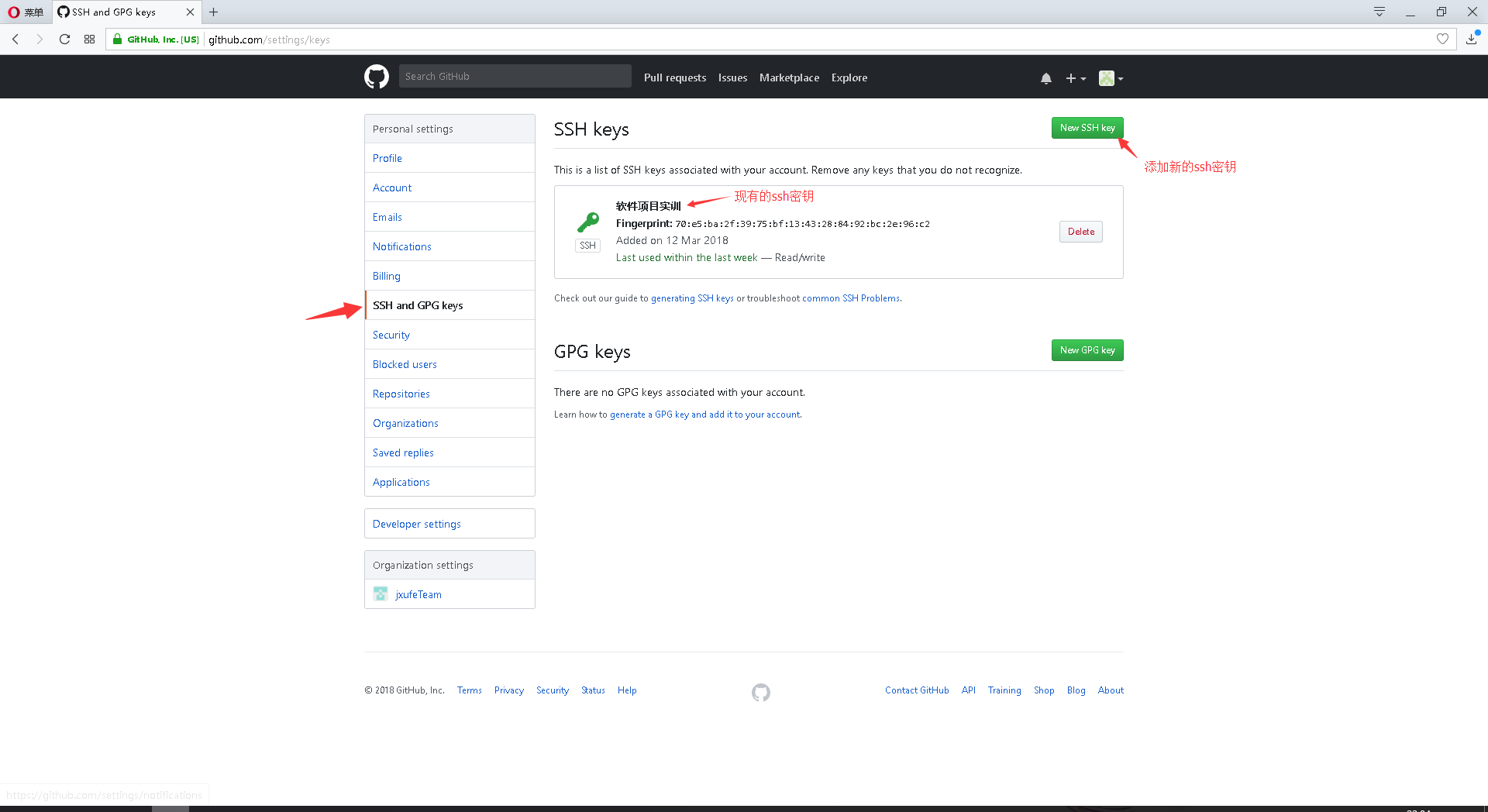
输入下面的指令创建ssh密钥文件，文件存放路径为电脑用户目录下的.ssh文件夹中

ssh-keygen -t rsa -C”你的GitHub的邮箱”



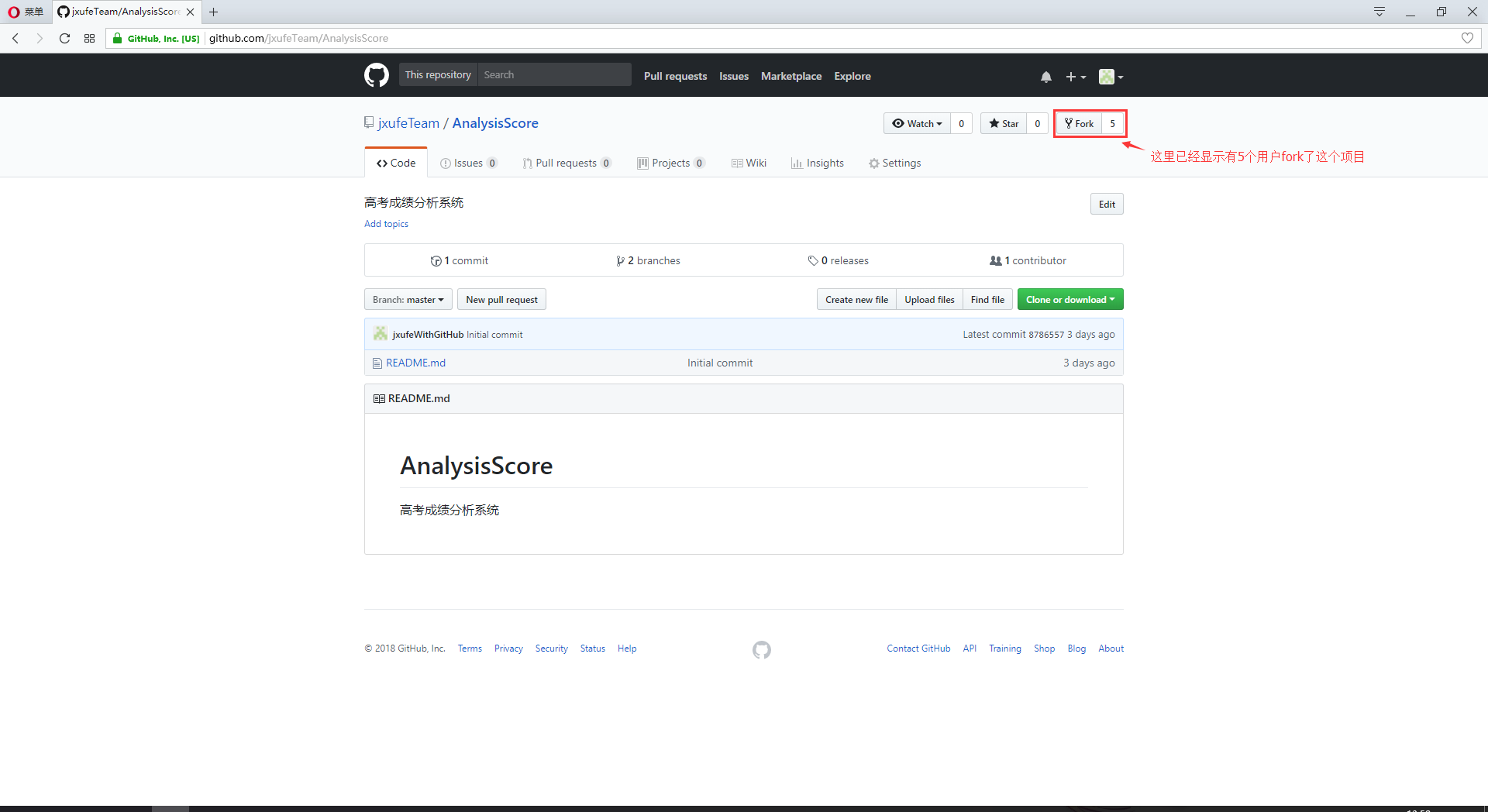


右键记事本方式打开这个.pub文件，复制其中内容。打开你的GitHub账号，在Settings中找到SSH and GPG keys，点击New SSH key，将刚刚复制的密钥粘贴保存即可

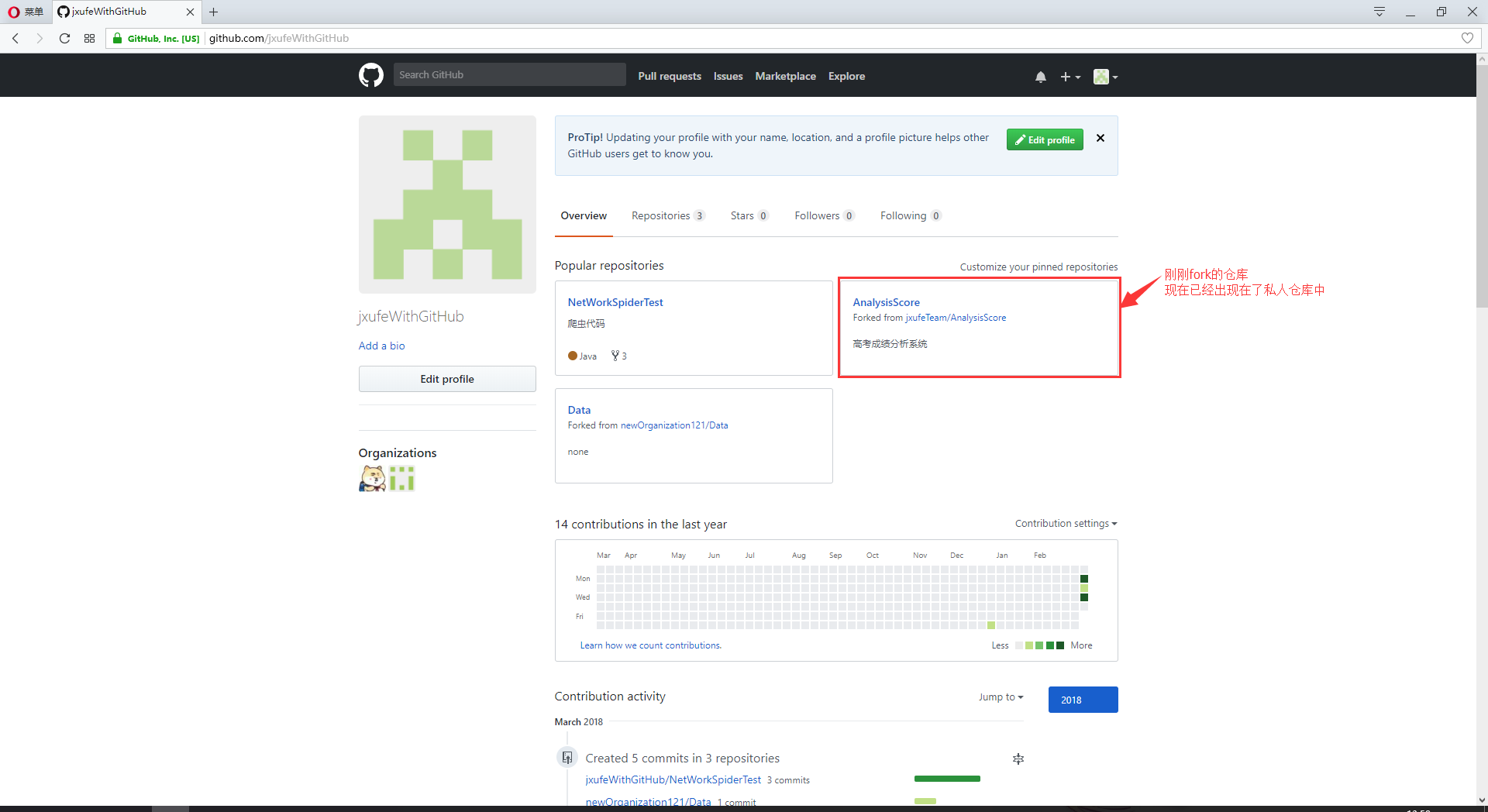


## 将团队仓库Fork到私人仓库

进入团队仓库项目中，点击下图中框选的Fork，即可将团队仓库fork到自己的私人仓库中

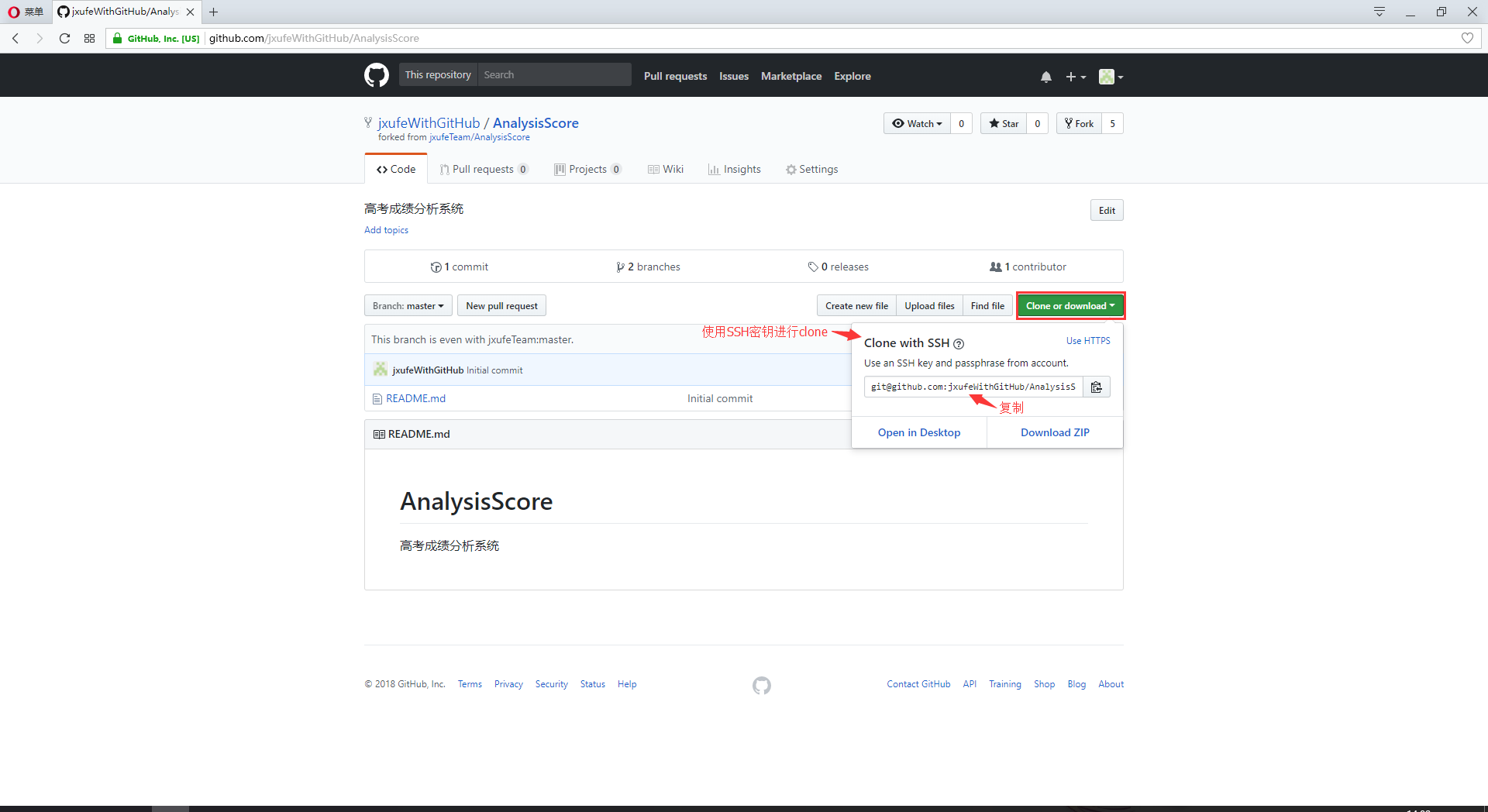


Fork完后，进入自己的profile中，可以查看到刚刚fork到私人仓库的项目



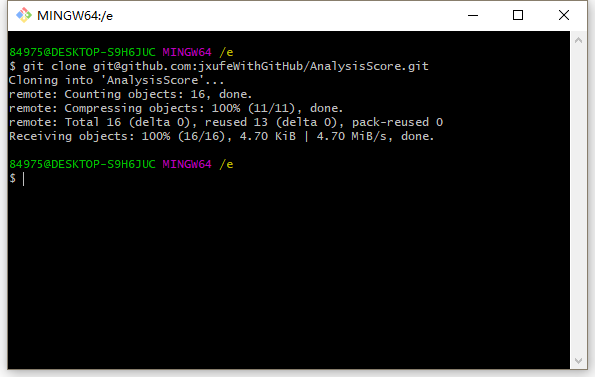
## 3.3 使用Git将私人仓库项目clone到本地仓库

打开私人仓库项目，点击Clone or download，选择Use SSH，复制地址



找到需要存放项目的路径，右键运行Git，输入下面的指令，将项目clone到本地

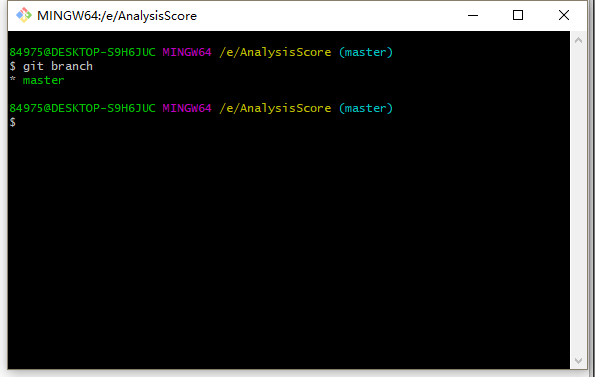
git clone 刚刚复制的地址



## 3.4 从master分支切换到dev分支

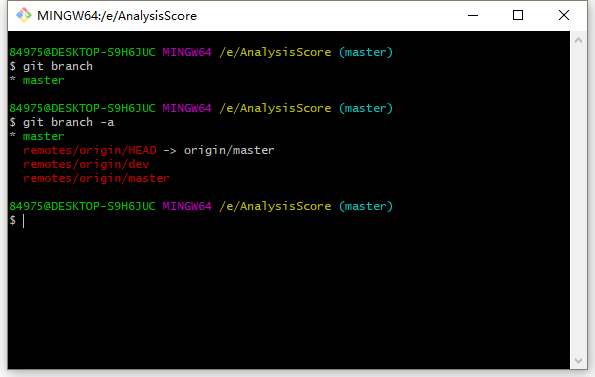
在刚刚clone的项目中右键打开Git，可以看到项目路径后面有个（master），此时我们输入下面的这条指令，查看我们的本地仓库有哪些分支

git branch



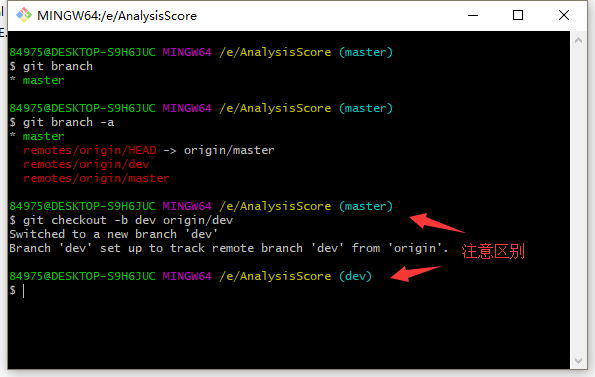
从上面可以看到，本地仓库只有master这一条分支，这是因为上面的指令是用来查看本地分支的。现在我们输入下面这条指令，查看本地仓库的远程分支

git branch -a

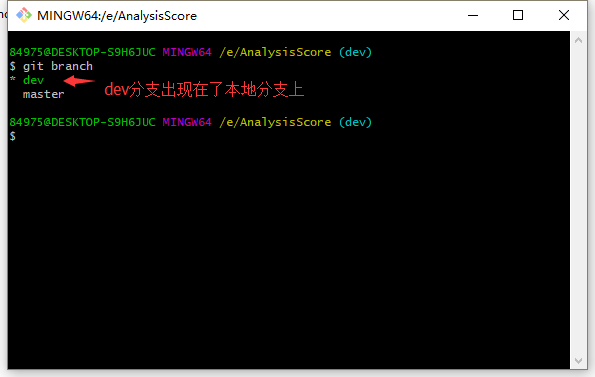


从上面可以看到，远程分支有三个，其中HEAD标记的是默认分支，dev分支和master分支都是我们GitHub项目上已存在的分支。前面已经说了，我们在开发项目时，一般是切换到dev（开发）分支上进行开发的，所以现在我们输入下面这条指令，将分支从master切换到dev

git checkout -b dev origin/dev

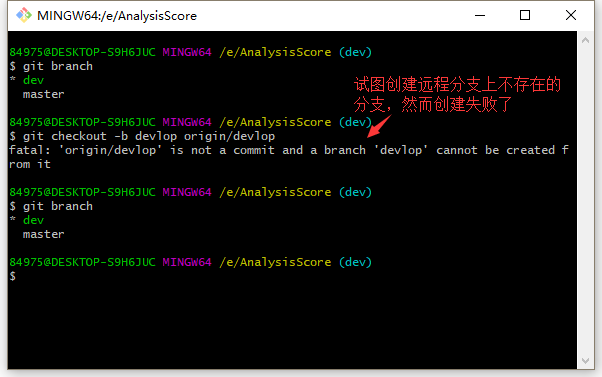


从上面可以明显看出，本地仓库已经从master分支切换到dev分支，那么我们刚刚输入的指令是什么含义呢？它的意思是，创建一个dev分支（-b），并把远程dev分支（origin/dev）的内容放在该分支内。接着切换到该分支（checkout）。那么现在我们再次输入查看本地分支的指令，看看有什么变化。



可以看到，我们远程分支上的dev分支已经出现在了本地分支上！

值得注意的是，如果你试图在本地创建一个远程分支上不存在的分支，操作会失败（如下图）

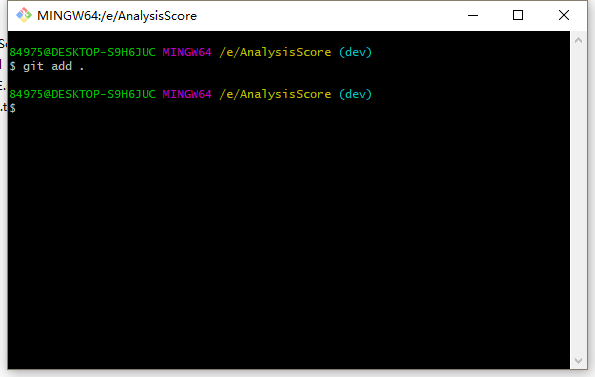


## 将新增的文件add到本地索引表中

我这里在刚刚clone下来的项目中添加了一个“团队成员.txt”文件，此时我们要将这个文件添加到本地索引表中，否则无法commit到本地仓库。

在项目目录下右键打开git，输入下面的指令进行add

git add .



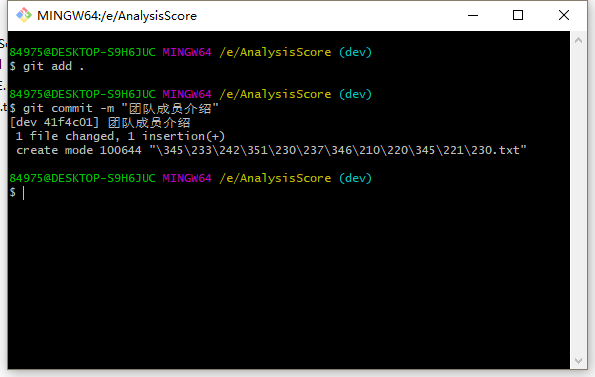
上面指令的含义是将新建的所有文件都添加到本地索引表中。

值得一提的是，如果我们要将本地的项目上传到空的GitHub仓库上，只需要将项目代码拷贝到刚刚clone下来的空仓库中，然后从本步骤开始向下执行即可上传。

## 将本地索引表中新增的文件commit到本地仓库中

执行完上一步的add操作后，就可以进行commit操作，将新建和更改的项目提交到本地仓库上，执行下面这段指令

git commit -m “注释”

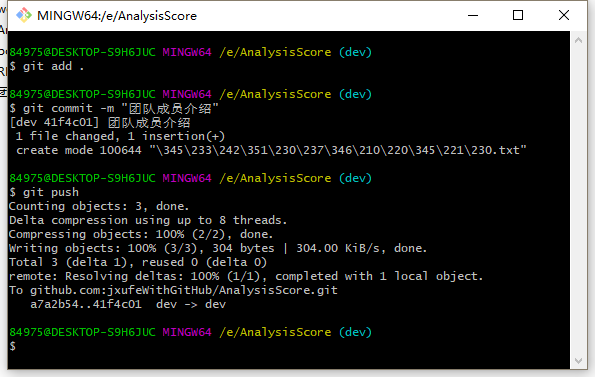


上面提示我们有一个文件改变并提交到本地仓库

## 将本地仓库push到GitHub的私人仓库中

在执行完上面的commit指令后，就可以执行push操作，将本地仓库修改到自己的GitHub仓库上了。执行下面这段指令

git push

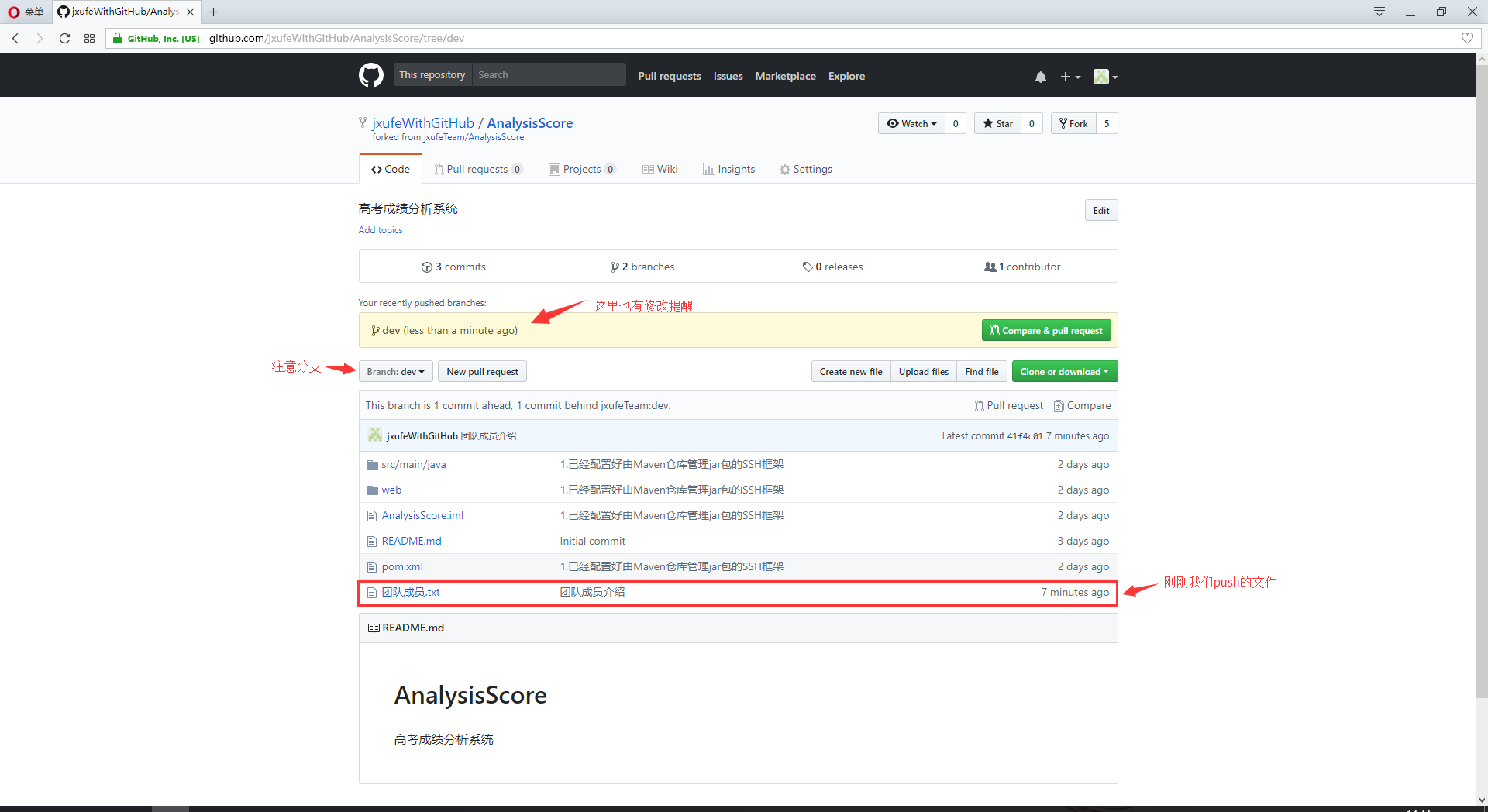


上面提示我们修改成功，已将本地仓库上的dev分支修改到GitHub仓库的dev分支上，随后你就可以在GitHub仓库上看到修改的文件与注释。

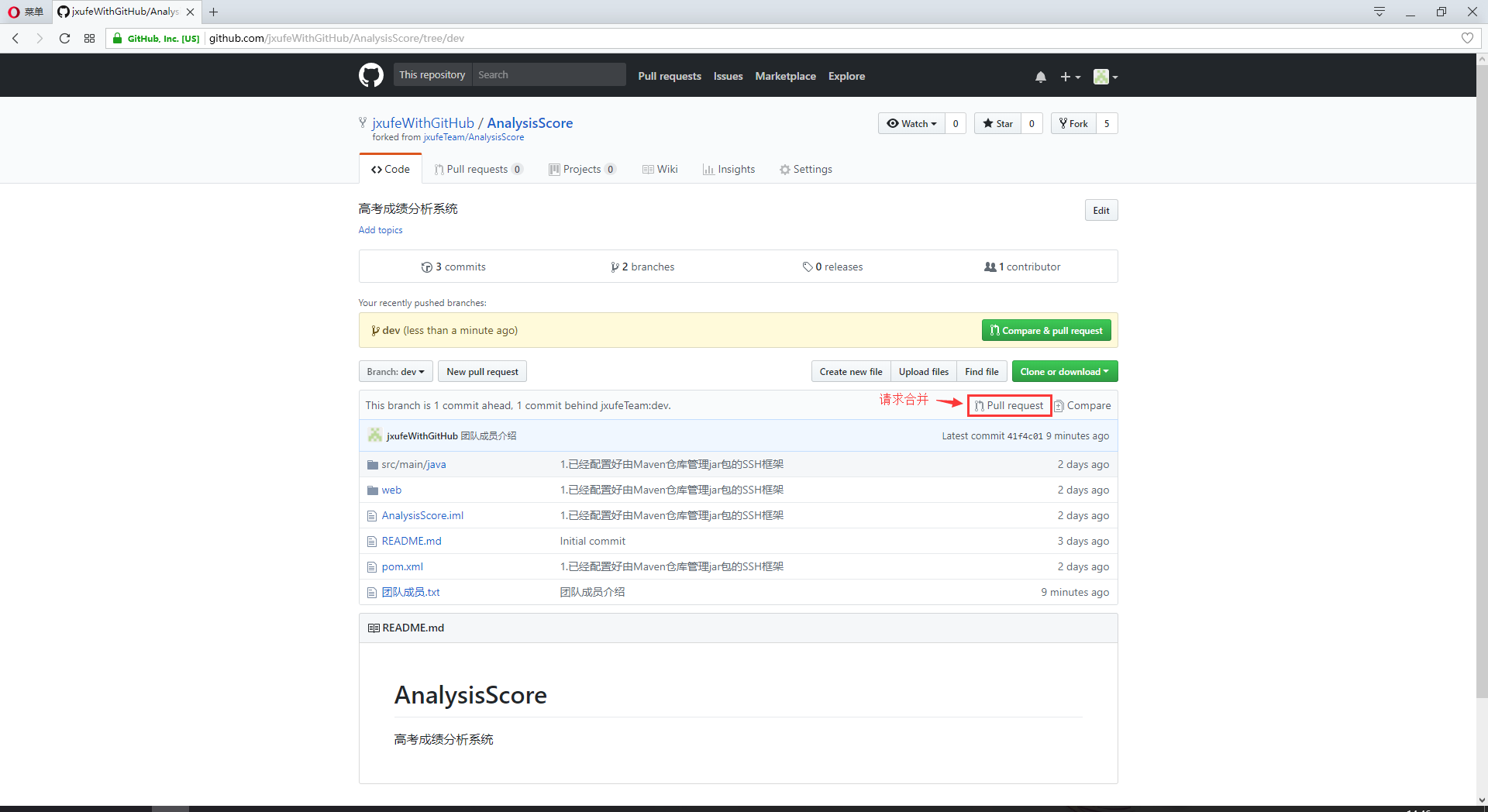
值得一提的是，刚刚我们执行的指令是不带任何参数的push指令，默认只修改当前分支。当然push指令可以带很多参数，这里不予以讲解，有兴趣可以在网上查看详细的push指令操作。

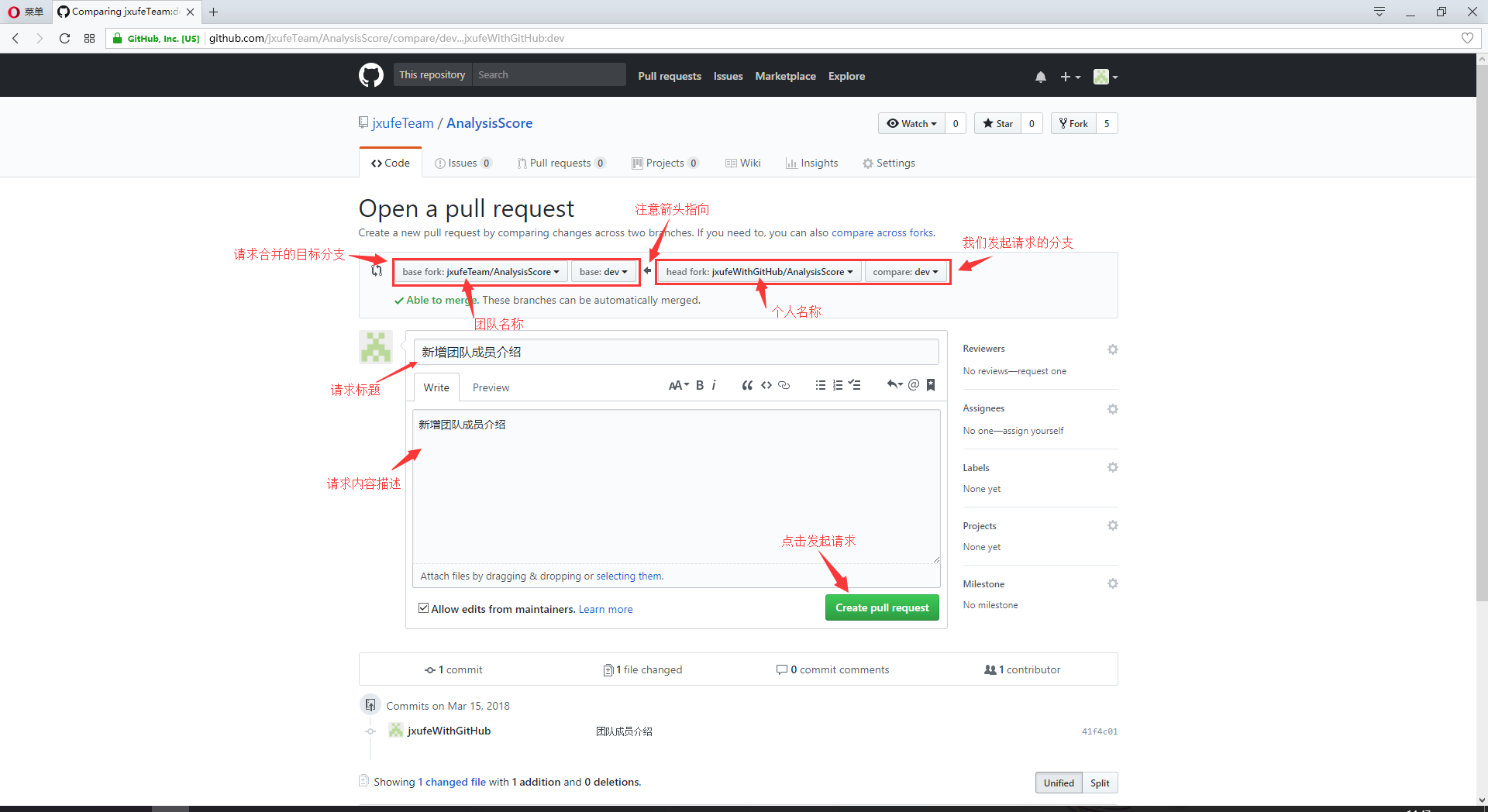
## 将私人仓库请求与团队仓库合并

现在我们回到GitHub上，打开我们的仓库，可以看到我们刚刚新增的文件和注释已经出现在了仓库上



点击Pull request，向团队仓库发起合并请求

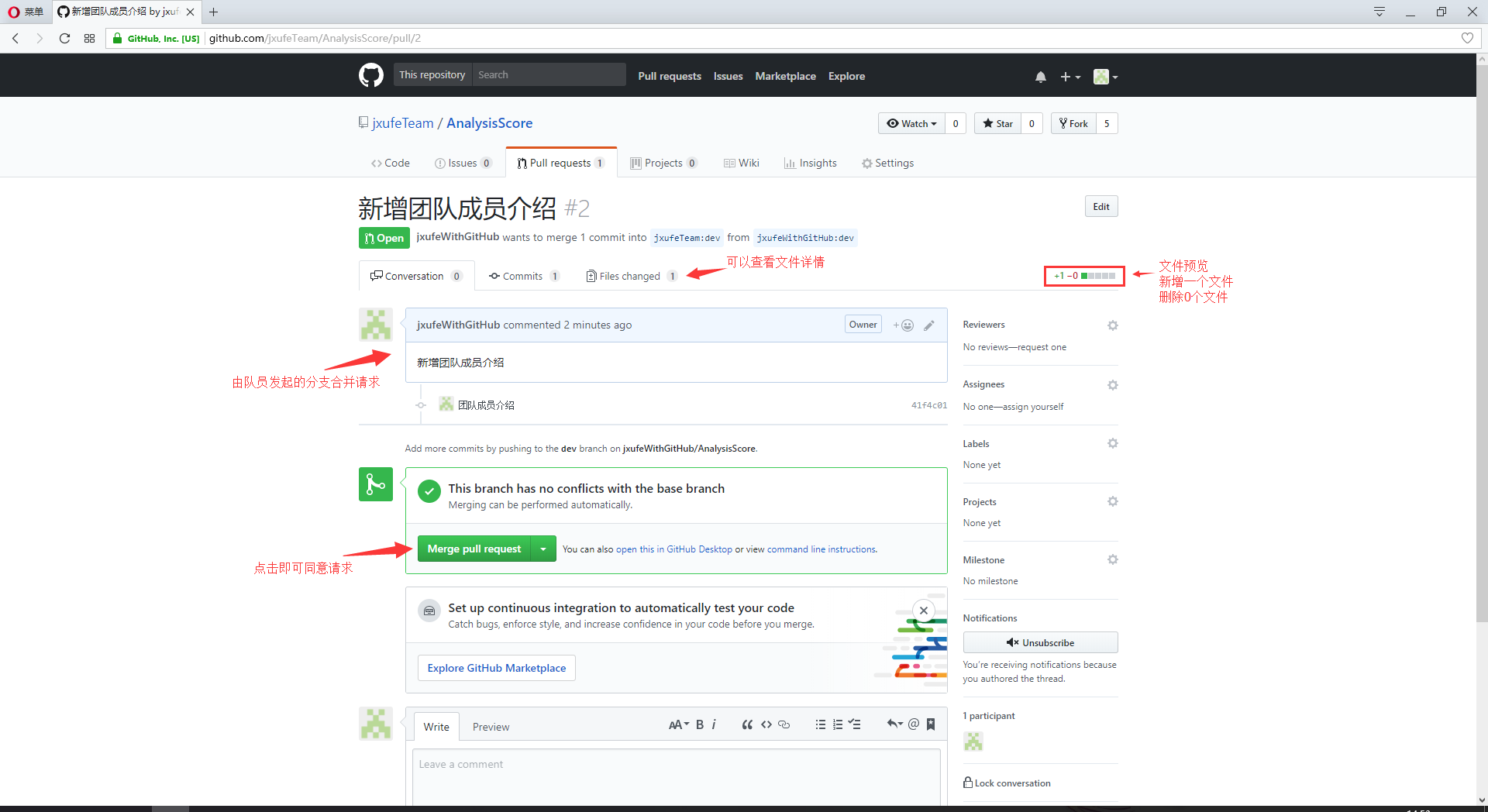




选择好请求分支和目标分支，并填写好请求标题和内容后，点击Create pull request即可发起请求。

## 管理员对合并申请进行审核

管理员可以查看请求内容，文件预览已经同意或不同意请求

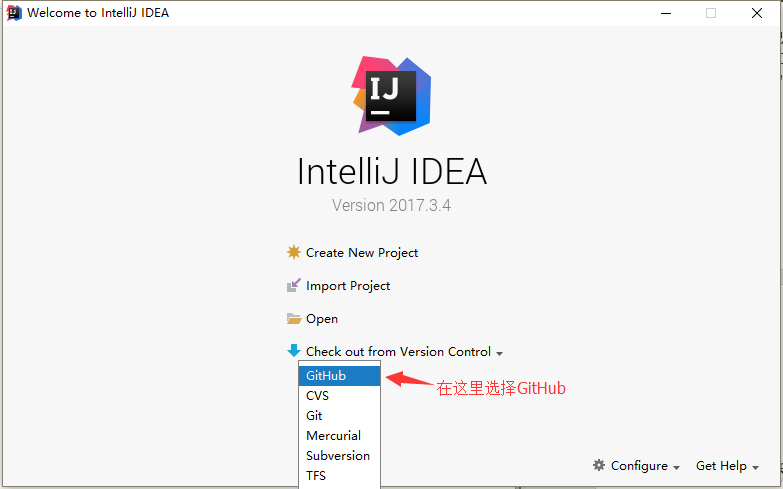


到这里，已经将基本的GitHub和Git的团队开发操作讲解完了，接下来我们来学习两种主流的开发工具（intellij与myeclipse）对于GitHub的使用。在下面的学习中，均默认已经配置了ssh密钥和fork了团队项目

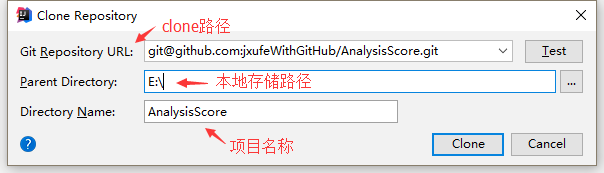
# 使用GitHub及Intellij进行团队开发

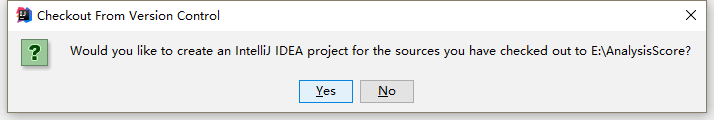
## 将私人仓库项目clone到本地仓库

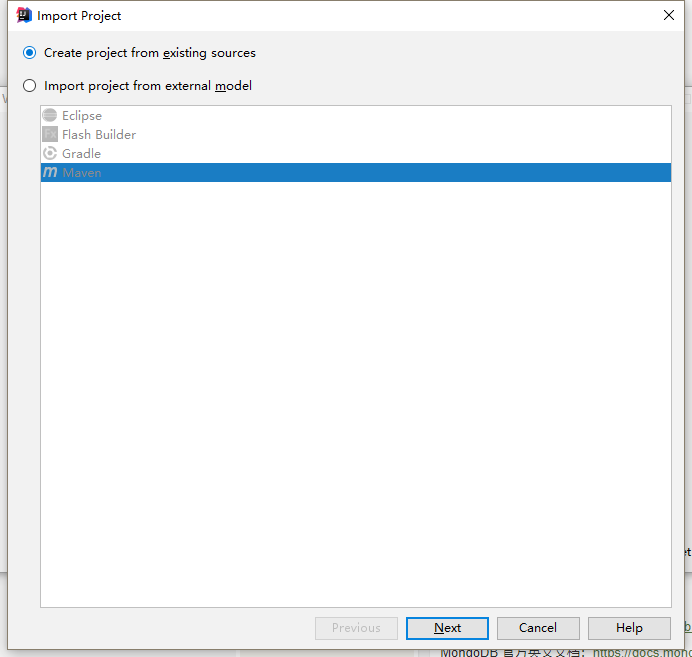
打开Intellij，选择从GitHub上clone项目，如下图



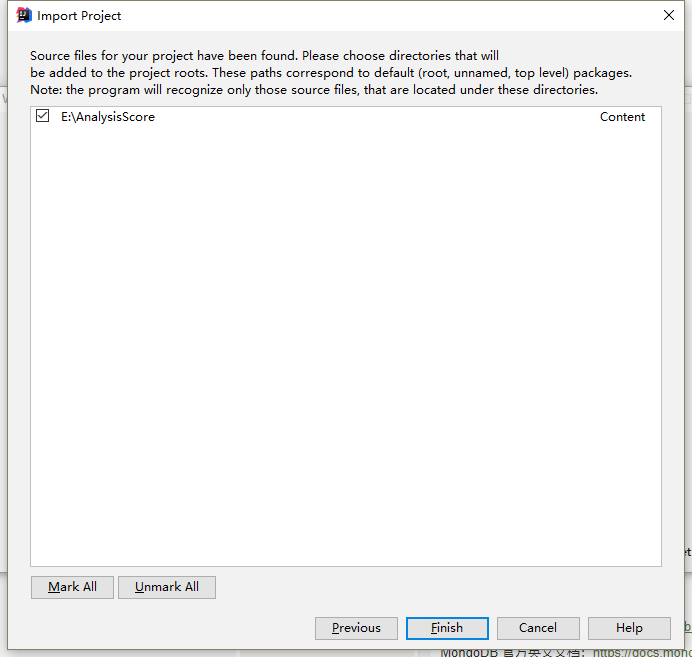
点击后弹出Clone Repository窗口，在URL中填写SSH地址，在Parent Directory中填写本地存储路径，Directory Name中填写项目名称，点击clone。

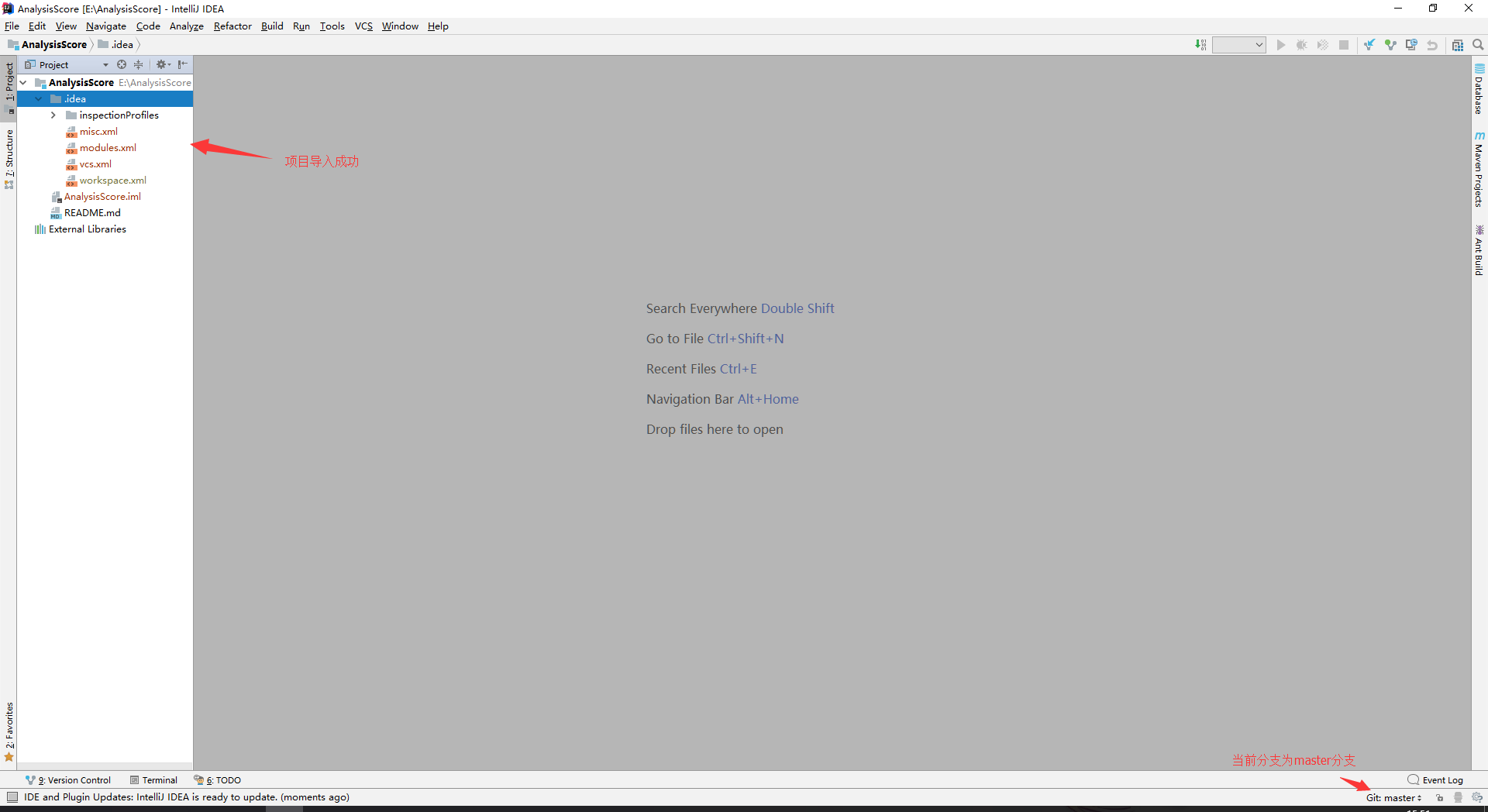






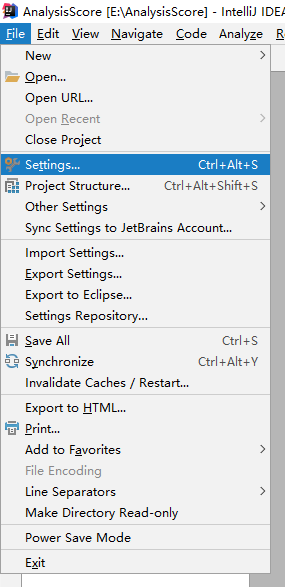


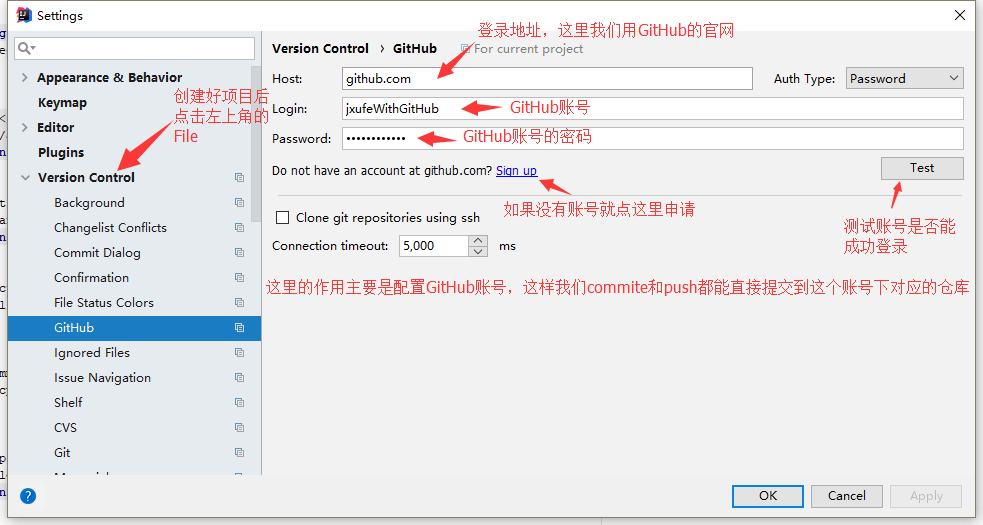


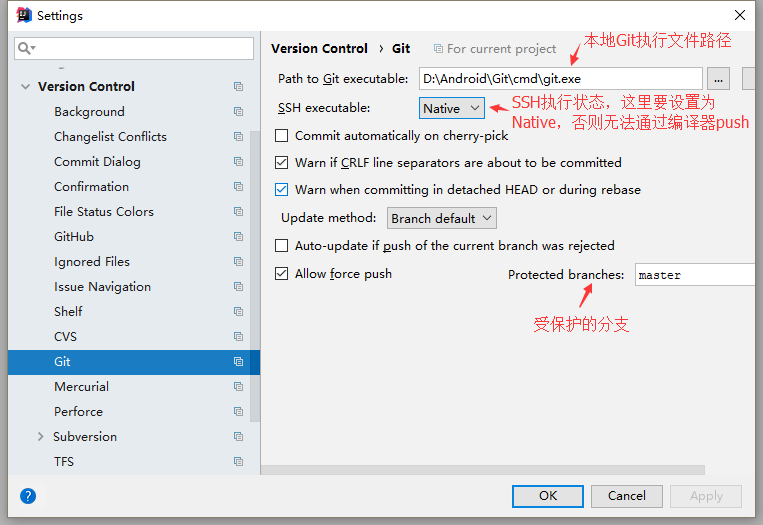


## 在Intellij中配置GitHub及Git

Clone完成后，需要在编译器中配置GitHub及Git，步骤如下图

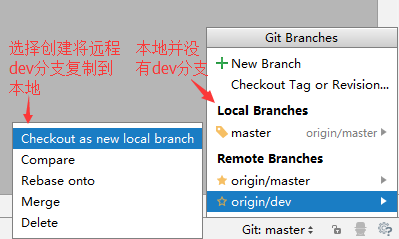






## master分支切换到dev分支

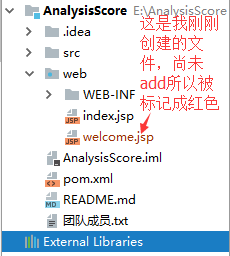
配置完成后，在编译器界面右下角有”Git:master”，点击将分支切换成dev，步骤如下图

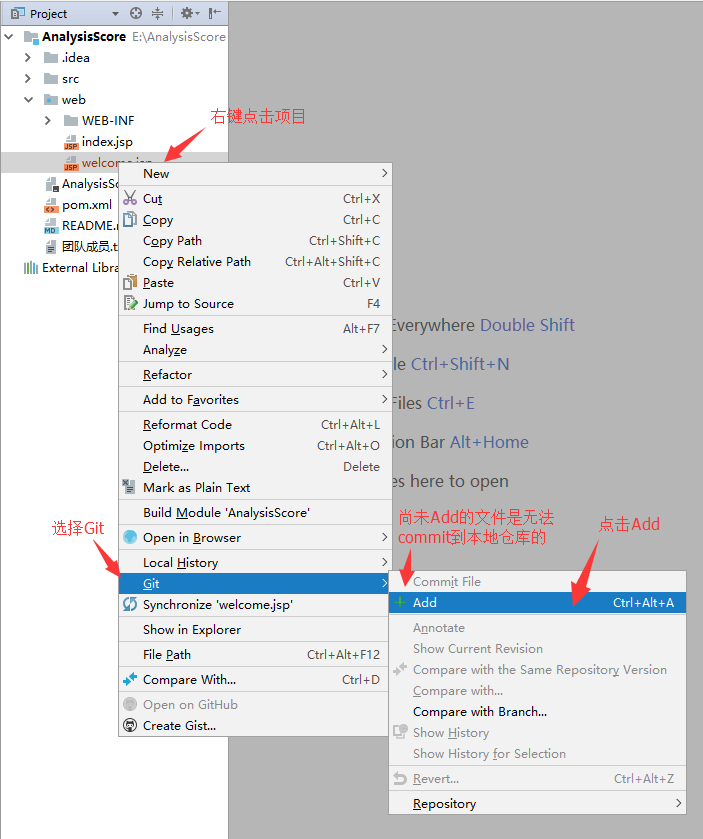


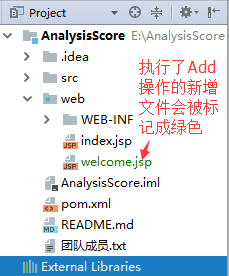
需要注意的是，如果有部分文件尚未add的话，这里是不允许切换分支的

## 将新增的文件add到本地索引表中

在Intellij中，仓库中原本存在的文件其文件名是黑色，新增的文件颜色若未执行add操作会被标记成红色，执行完add操作后则会变成绿色，而修改的文件的文件名会被标记成蓝色。在这里我们只需要将被标记成红色的文件进行add操作即可，步骤如下图。

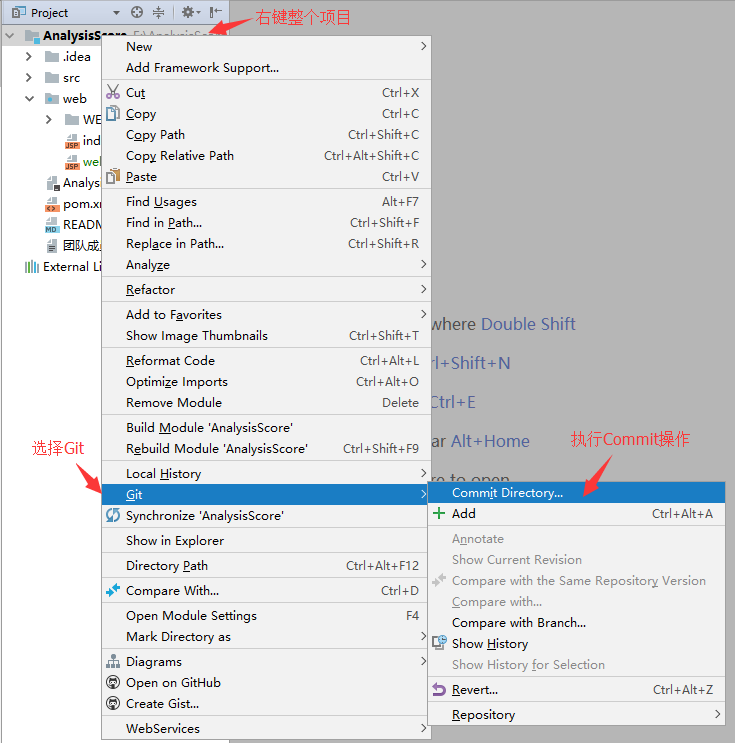


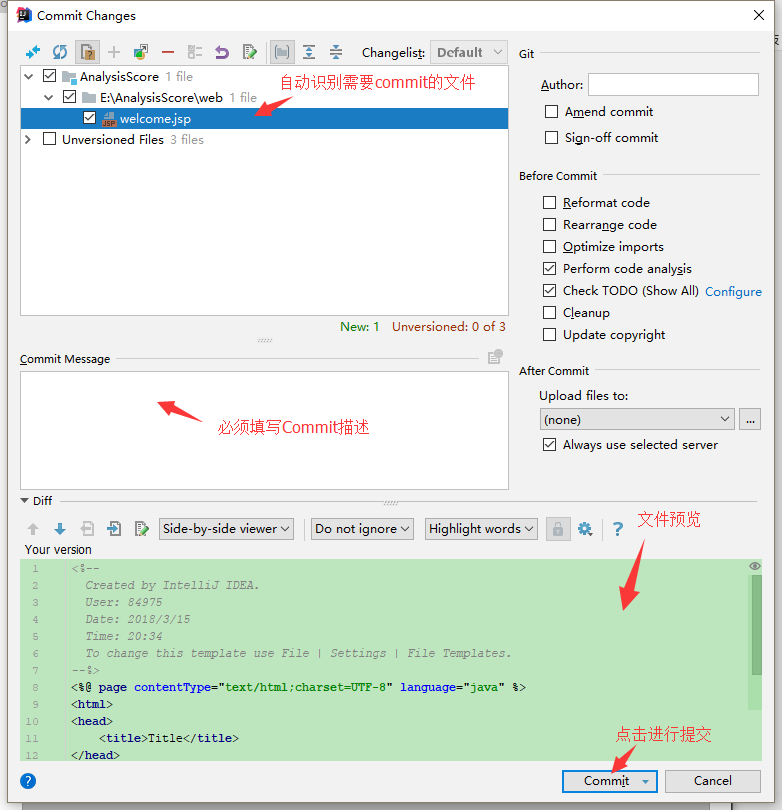


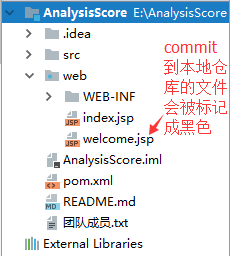


## 将本地索引表中新增的文件commit到本地仓库中

如果你新增或修改的文件很多的话怎么办？难道要一个一个commit到本地仓库上？在Intellij中，我们可以直接右键点击整个项目进行commit操作，编译器会自动提取出需要commit的文件给你预览，在这里你就可以选择需要commit进行提交了，步骤如下图。

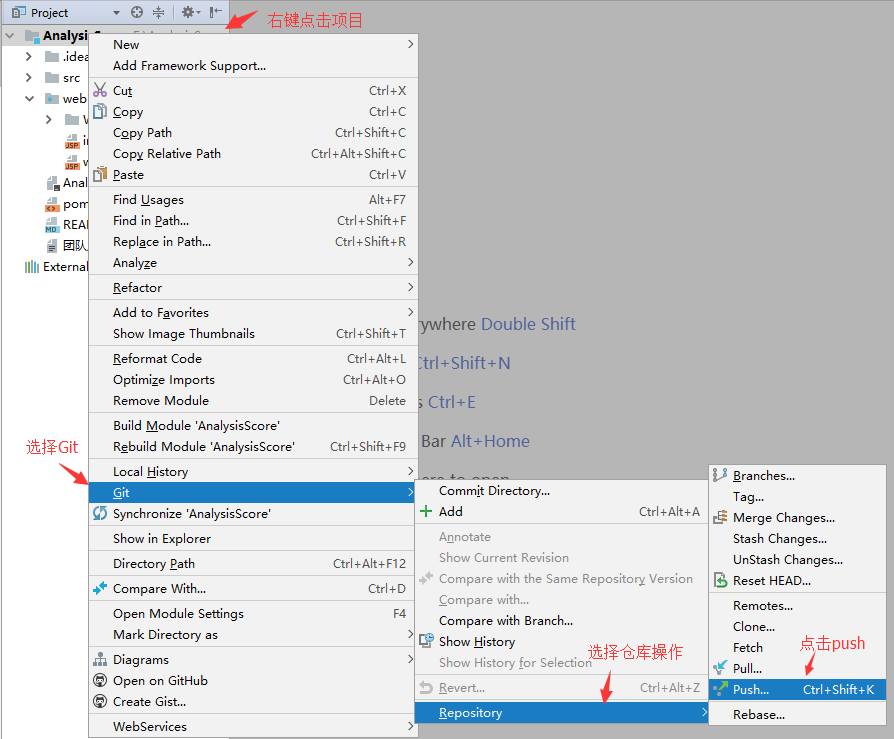


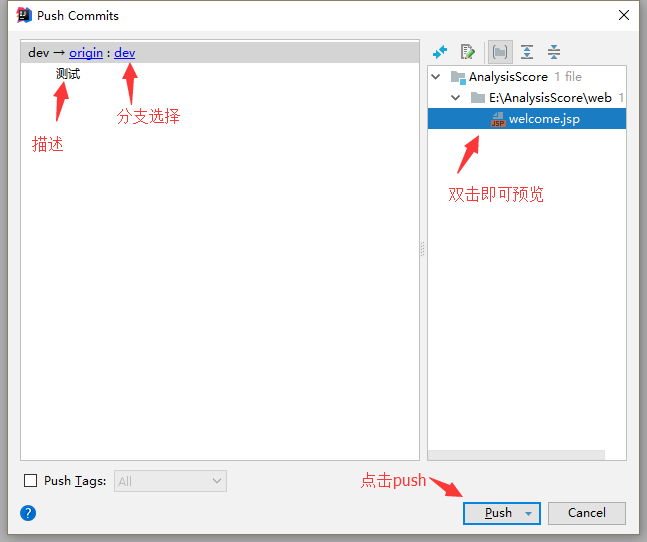


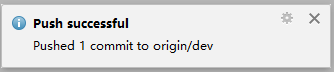


## 将本地仓库push到GitHub的私人仓库中

在确认所以需要push的文件都已经commit，可以开始执行本步骤的操作，操作如下。







执行完成后，你就可以在你的GitHub看到提交的文件了

后面的请求合并和管理员审核操作不变

# 使用GitHub及MyEclipse进行团队开发

由于电脑环境限制，没有安装 MyEclipse，所以很遗憾没有关于MyEclipse的使用讲解