# Day15

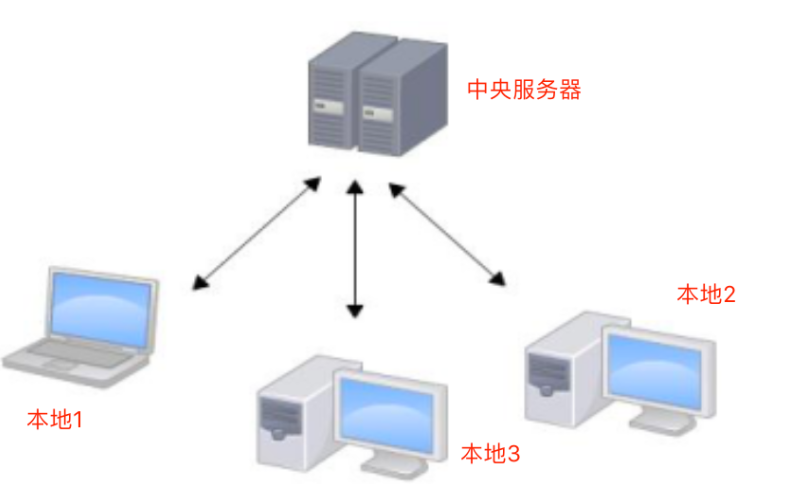
## 今日主要内容

1. Git介绍
2. Git的特点
3. Git的下载和安装
4. Git的基本命令
5. 本地仓库的相关操作
6. 分支管理
7. 远程仓库
8. 同步远程仓库
9. 远程仓库的操作
10. Idea中集成Git
11. Idea的Git操作
12. Idea操作分支
13. 回退历史版本

### Git介绍

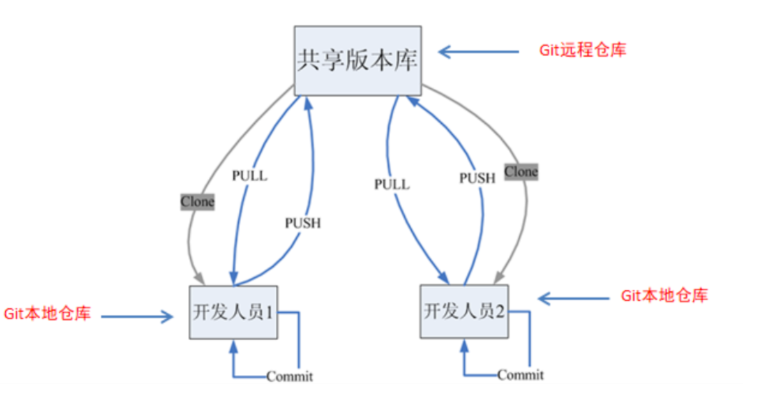
Git是一个开源的分布式版本控制系统(Distributed Version Control System，简称DVCS)。可以有效、高速地处理从很小到非常大的项目版本管理。

版本控制系统能追踪项目，从开始到结束的整个过程。对编程人员而言，版本控制技术是团队协作开发的桥梁，助力于多人协作同步进行大型项目开发。软件版本控制系统的核心任务:查阅项目历史操作记录、实现协同开发。



分布式版本控制工具

分布式版本控制系统可以没有 “中央服务器”，每个人的电脑上都是一个完整的版本仓库，这样工作的时候，不需要联网。因为版本仓库就在你自己的电脑上。多人协作只需要各自修改，开发完成即可，推送给对方【联网】，推送的时候是将整个版本仓库推过去。如：Git



### Git的特点

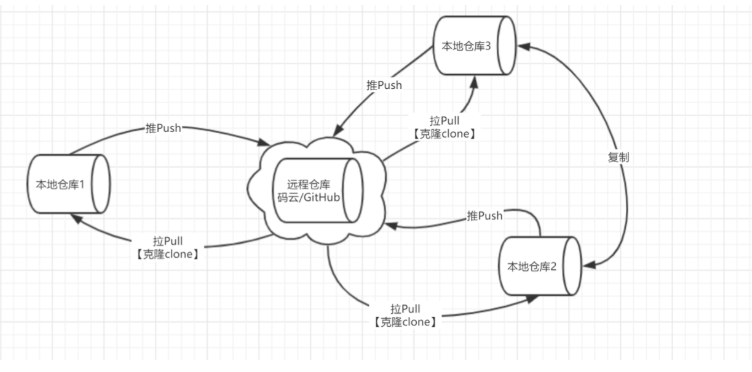
① 速度、简单的设计

② 对非线性开发模式的强力支持(允许成千上万个并行开发的分支)

③ 完全分布式

④ 有能力高效管理类似 Linux 内核一样的超大规模项目(速度和数据量)

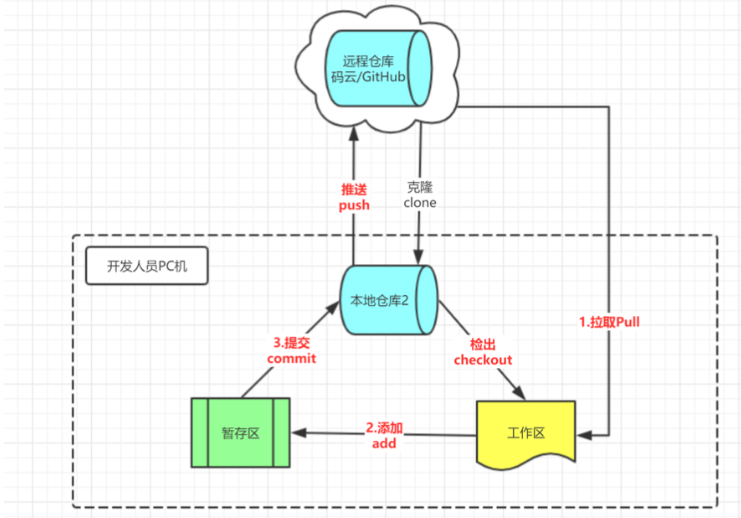
⑤ 协同开发



Clone:克隆，从远程仓库中克隆代码到本地仓库，第一次操作

Push:推送，代码完成后，需要和团队成员共享代码时，将代码推送到远程仓库。

Pull:拉取，从远程库拉代码到本地库，自动进行合并(merge)，最后放到工作区。



checkout:将本地仓库的内容检出到工作区

add:在提交前先将代码提交到暂存区

commit:提交到本地仓库

本地仓库: 在本地主机上的一个代码库，可以独立存在，也可以与远程仓库进行关联

工作区:对任何文件的修订(增删改)，都先放在工作区，工作区不与任何仓库分支进行关联 暂存区:把修订的文件，从工作区经过add(添加)后与某一个仓库分支进行关联，只要进 入缓存区的文件才能commit(提交)到本地仓库。

远程仓库 : 在局域网或互联网上的一个主机，存放代码库的主机或平台，比如GitHub、 Gitee.com(码云)

分支:代码存放在仓库，默认是主分支(master)，可以在主分支基础上创建很多子分支，比如 develop(开发)、bugfix(bug修复)等。

### Git的下载和安装

下载地址：<https://git-scm.com/download>





傻瓜式安装：一路下一步。安装完成后在电脑桌面右击显示

注：

Git GUI Here：Git提供的图形界面工具

Git Bash Here：Git提供的命令行工具



### Git的基本命令

1.安装完成 Git 后，正式使用git前，是需要进行一些全局设置的，如用户名、邮箱。

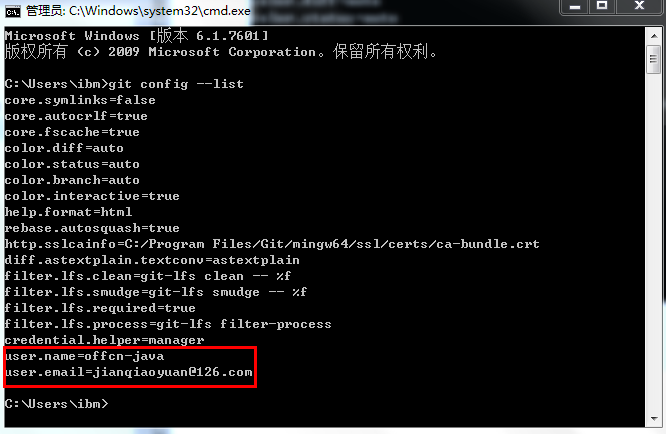
# 设置全局用户名

git config --global user.name "your name"

# 设置邮箱

git config --global user.email "your email"

1. 查看配置信息：git config --list



3. 构建本地仓库：要使用Git对我们的代码进行版本控制，首先需要构建本地仓库

① 在本地初始化一个Git仓库

② 从远程仓库克隆一个仓库

### 地仓库的相关操作

#### 1.在本地初始化一个Git仓库:

- 在电脑的任意位置创建一个**空目录**作为我们的本地Git仓库

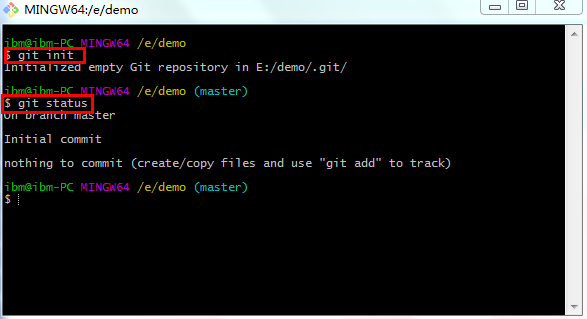
- 进入这个目录中，点击右键打开Git bash窗口

- 执行命令Git init

如果在当前目录中看到.git文件夹(此文件夹为隐藏文件夹)则说明Git仓库创建成功

#### 2. 查看文件状态：

git status [-s]



#### 3. 将文件添加(修改)到版本库

要将一个文件纳入到版本库管理，首先要将其添加到暂存区，然后才能提交到仓库中。

# 添加单个文件到暂存区

git add Readme.txt

# 将当前目录下所有修改添加到暂存区，除按照规则忽略的之外

git add .



#### 4. 将暂存区中的文件，提交到仓库中

# 如果暂存区有文件，则将其中的文件提交到仓库

git commit

# 带评论提交，用于说明提交内容、变更、作用等

git commit -m 'your comments'





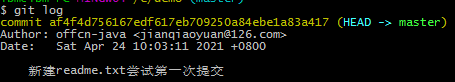
#### 5. 查看提交历史记录

有的时候，是会需要查看自己做过哪些提交，来回顾自己完成的部分。或者需要寻找某个具体的提交来

查看当时的代码。

git log # 显示所有提交的历史记录

git log --pretty=oneline # 单行显示提交历史记录的内容



修改readme.txt的内容





把readme.txt添加到暂存区

git add readme.txt

git commit -m ‘第二次提交’



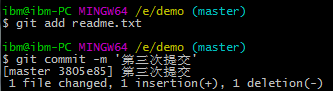
再次修改readme.txt的内容

hello this is readme document.-demo1-demo2

把readme.txt添加到暂存区

git add readme.txt

git commit -m ‘第三次提交’



查看提交日志



单行显示提交历史记录的内容

git log --pretty=oneline



#### 6. 版本回退

有了 git log 来查看提交的历史记录，我们就可以通过 git reset --hard 来回退到我们需要的特定版本，然后使用当时的代码进行各种操作。

# 回退到 commit\_id 指定的提交版本

git reset --hard 'commit\_id'

回退到第二次提交

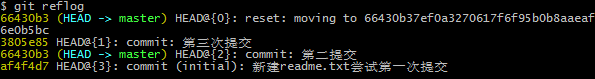


#### 7. 回到以前的某个提交

当退回到某个提交的版本以后，再通过 git log 是无法显示在这之后的提交信息的。但是，通过 git reflog 可以获取到操作命令的历史。因此，想要回到未来的某个提交，先通过 git reflog 从历史命令中找到想要回到的提交版本的 ID， 然后通过 git reset --hard 来切换。

git reflog

git reset --hard 'commit\_id'





#### 8. 删除文件

在文件未添加到暂存区之前，对想删除文件可以直接物理删除。如果文件已经被提交，则需要 git rm来删除。

git rm Readme.txt // 删除已经被提交过的 Readme.txt

注意: git rm 只能删除已经提交到版本库中的文件。其他状态的文件直接用这个命令操作是出错的。



### 分支管理

#### 查看分支

# 查看本地分支信息

git branch



# 查看相对详细的本地分支信息

git branch -v



# 查看包括远程仓库在内的分支信息

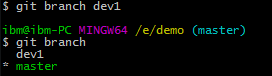
git branch -av

注意：前面带有\*号，这标识我们当前所在的分支

#### 创建分支

# 新建一个名称为 dev 的分支

git branch dev

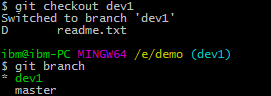


#### 切换分支

# 新建完 dev 分支以后，通过该命令切换到 dev 分支

git checkout dev1

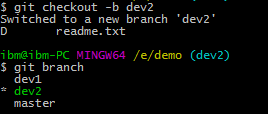
注意：当我们创建完分支以后，我们需要切换到新建的分支，否则，所有的修改，还是在原来的分支上。事实上，所有的改动，只能影响到当前所在的分支。



#### **创建并切换分支**

# 新建 dev 分支，并切换到该分支上

git checkout -b dev



#### **合并分支**

# 切换回 master 分支

git checkout master

# 将 dev 分支中的修改合并回 master 分支

git merge dev

注意：分支修改文件中如果有换行的话会报错

解决：Git默认配置替换回车换行成统一的CRLF，我们只需要修改配置禁用该功能即可。

git config --global core.autocrlf false

#### **删除分支**

当之前创建的分支，完成了它的使命，如 Bug 修复完，分支合并以后，这个分支就不在需要了，就可 以删除它。

# 删除dev分支

git branch -d dev1

### 远程仓库

现在我们已经在本地创建了一个Git仓库，又想让其他人来协作开发，此时就可以把本地仓库同步到远 程仓库，同时还增加了本地仓库的一个备份。那么我们如何搭建Git远程仓库呢?我们可以借助互联网上提供的一些代码托管服务平台来实现，其中 比较常用的有GitHub、码云等。

-- GitHub( 地址:https://github.com/ )是一个面向开源及私有软件项目的托管平台，因为只支持 Git 作为唯一的版本仓库格式进行托管，故名GitHub。

-- 码云(地址: https://gitee.com/ )是国内的一个代码托管平台，由于服务器在国内，所以相比于 GitHub，码云速度会更快。

接下来我们演示如何将本地仓库中的代码同步到github。和码云的操作一模一样.

#### 注册账号

1. 第一步，点击注册按钮





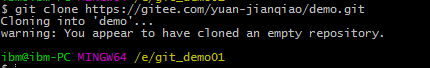
#### **登录使用**

### **同步远程仓库**

Github支持两种同步方式“https”和“ssh”。如果使用https很简单基本不需要配置就可以使用，但是每次 提交代码和下载代码时都需要输入用户名和密码。而且如果是公司配置的私有git服务器一般不提供 https方式访问，所以我们要来着重演示“ssh”方式。

#### https克隆

命令：git clone https地址



#### ssh配置

SSH是英文Secure Shell的简写形式。通过使用SSH，你可以把所有传输的数据进行加密，这样"中间 人"这种攻击方式就不可能实现了，而且也能够防止DNS欺骗和IP欺骗。使用SSH，还有一个额外的好处 就是传输的数据是经过压缩的，所以可以加快传输的速度。

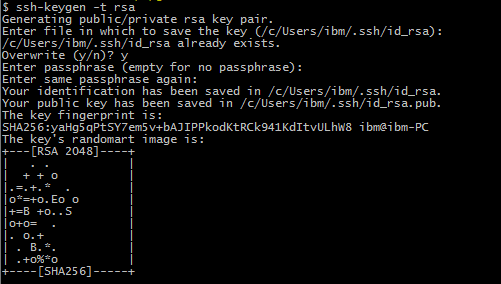
注：使用SSH同步方式需要先生成密钥并在GitHub配置公钥。

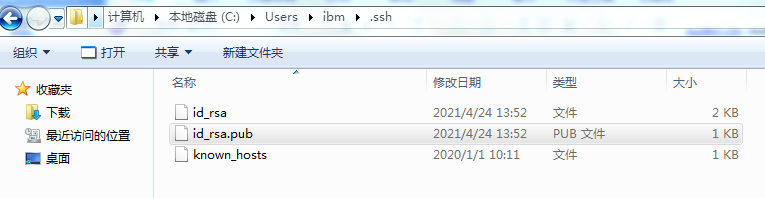
执行命令,生成公钥和私钥:

ssh-keygen -t rsa

执行命令完成后,在window本地用户.ssh目录C:\Users\用户名.ssh下面生成如下名称的公钥和私钥。

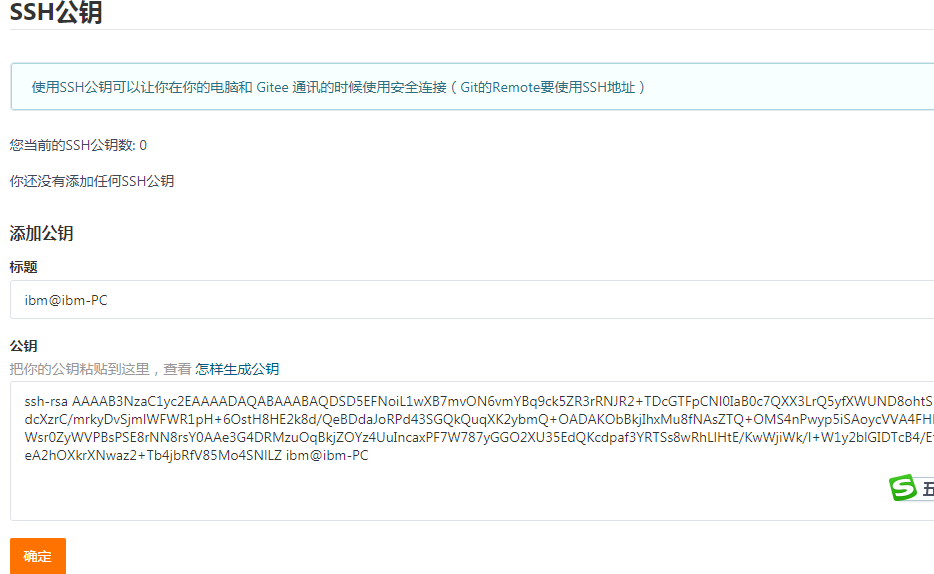
密钥生成后需要在github上配置密钥，本地才可以顺利访问。

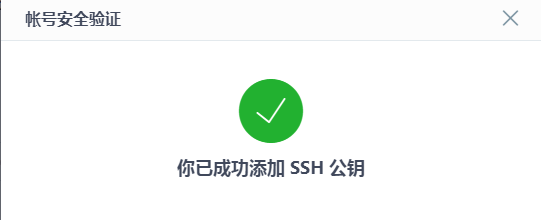




打开gitee中设置添加公钥







### **远程仓库的操作**

#### **查看远程仓库**

# 命令形式:

git remote -v

# origin ——仓库服务器的默认名称

注：如果显示空，没有添加远程仓库。如果添加显示远程仓库地址



#### **添加远程仓库**

如果已经有了一个本地仓库，然后打算将它发布到远程，供其他人协作。

那么使用命令：

# 为本地仓库添加远程仓库

git remote add origin 远程仓库地址

如：git remote add origin https://gitee.com/junguang00/git\_demo01.git

注：

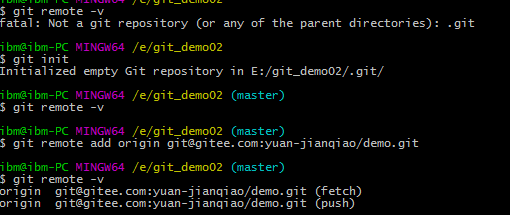
提示出错信息：fatal: remote origin already exists.

先输入

$ git remote rm origin

再输入：

git remote add origin 远程仓库地址



#### **推送本地内容到远程仓库# 在本地仓库更新内容 添加暂存区**

git add .

# 提交内容

git commit -m '信息'

# 当本地仓库中，代码完成提交，就需要将代码等推送到远程仓库，这样其他协作人员可以从远程仓库同步内容。

# 第一次推送时使用，可以简化后面的推送或者拉取命令使用

git push -u origin master

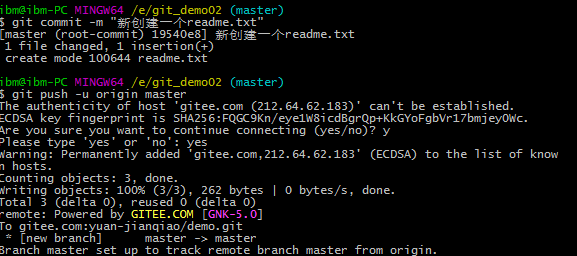
# 将本地 master 分支推送到 origin 远程分支

git push origin master

注意:

① git push -u origin master ，第一次使用时，带上 -u 参数，在将本地的 master 分支推送 到远程新的 master 分支的同时，还会把本地的 master 分支和远程的 master 分支关联起来。

② 推送之前，需要先pull远端仓库，如果发现提交版本不一致，出现错误



#### 拉取仓库数据

在多人协作过程中，当自己完成了本地仓库中的提交，想要向远程仓库推送前，需要先获取到远程仓库的最新内容。使用如下命令

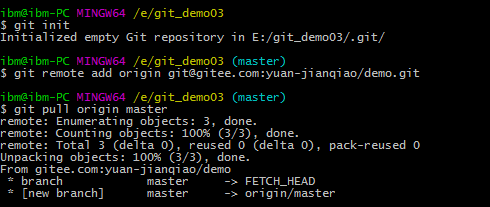
git fetch origin master

git pull origin master

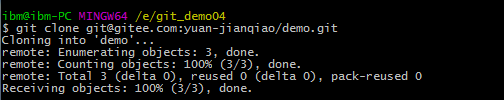
git fetch 和 git pull 之间的区别:

git fetch 是仅仅获取远程仓库的更新内容，并不会自动做合并。

git pull 在获取远程仓库的内容后，会自动做合并，可以看成 git fetch 之后 git merge 。



如果在本地新创建一个目录不是仓库则不能使用pull来拉取，应使用clone进行克隆



#### **移除无效的远程仓库**

# 命令形式:

git remote rm <shortname>

注意:此命令只是从本地移除远程仓库的记录，并不会真正影响到远程仓库



#### **解决合并冲突**

在一段时间，A、B用户修改了同一个文件，且修改了同一行位置的代码，此时会发生合并冲突。

例如：

A用户在本地修改代码后优先推送到远程仓库，此时B用户在本地修订代码，提交到本地仓库后，也需 要推送到远程仓库，此时B用户晚于A用户推送，故需要先拉取远程仓库代码，经过合并后才能推送代 码。在B用户拉取代码时，因为A、B用户同一段时间修改了同一个文件的相同位置代码，故会发生合并 冲突。

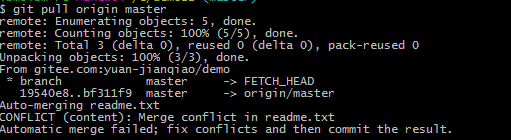
解决：

① 先拉取代码

② 打开代码解决冲突

③ 再提交

1.重新拉取最新数据

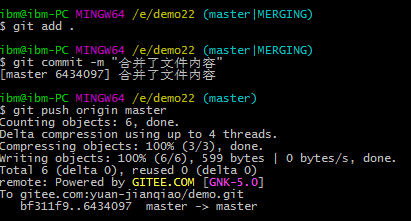


2.修改文件内容

这是一个readme.txt文件-demo22

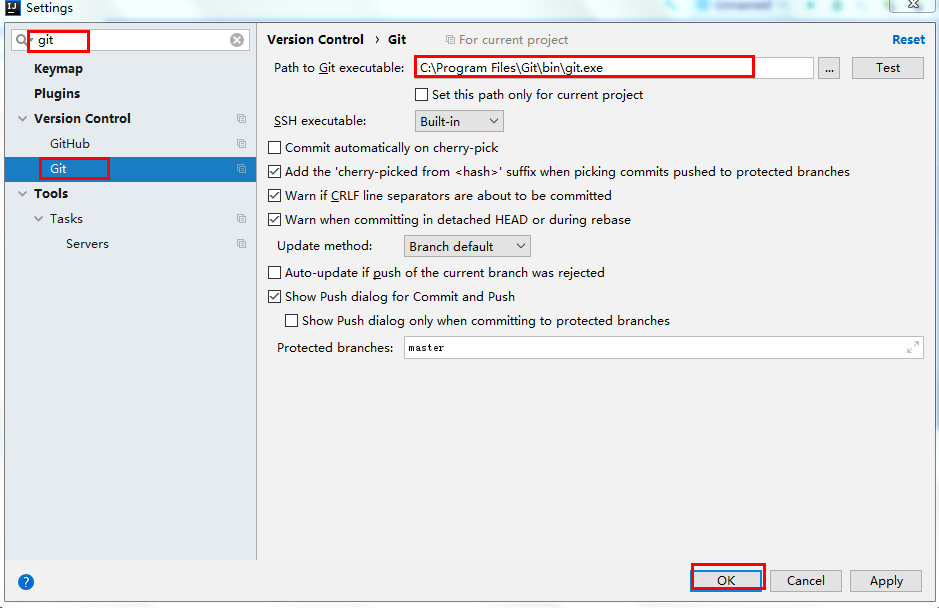
这是一个readme.txt文件-demo11

3.重新推送到远程仓库的主分支上



### Idea中集成Git

1. 打开File-->settings-->搜索git并指定git的按装路径



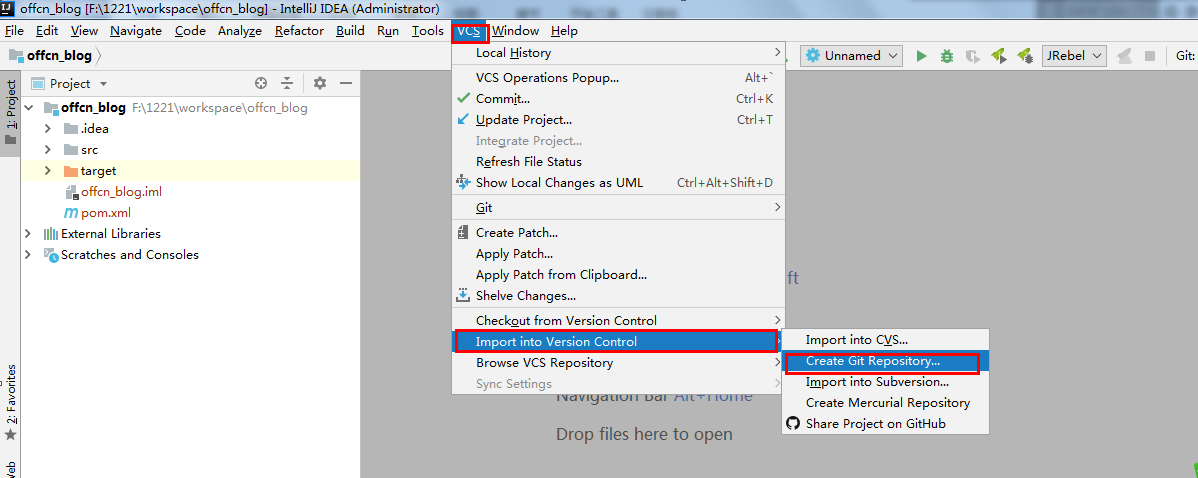
### Idea的Git操作

#### 在码云上创建一个仓库



将工程交给Git管理

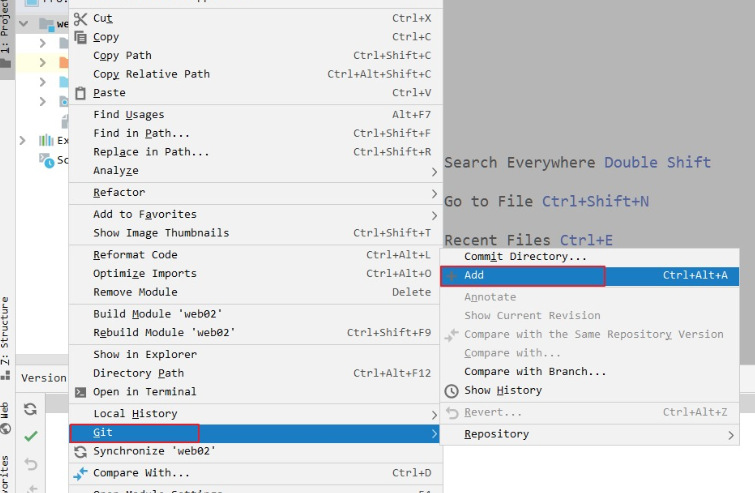
打开VCS--->import into Version Control-->cdreate Git Respositoy把项目交给git管理



选项目的上一层为仓库

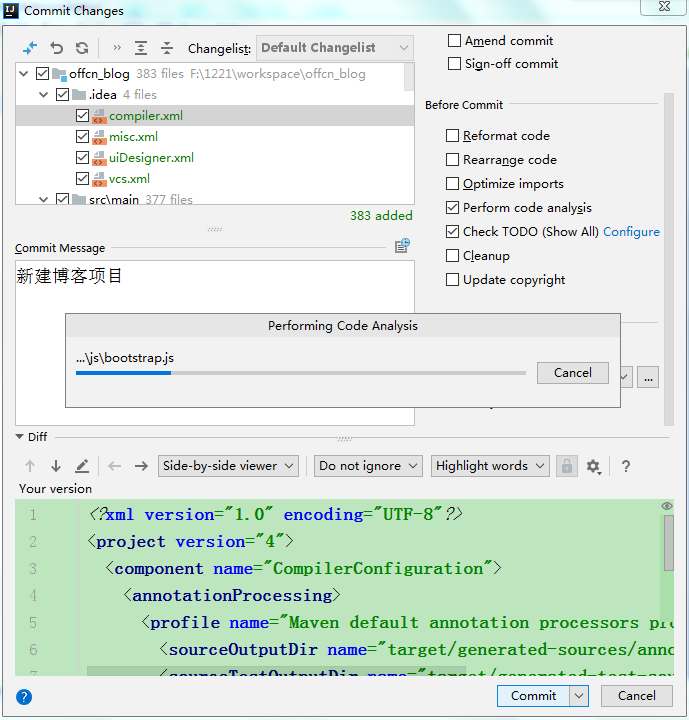
#### 提交到本地仓库

在项目上右键，选择git -->add 把项目添加到暂存区

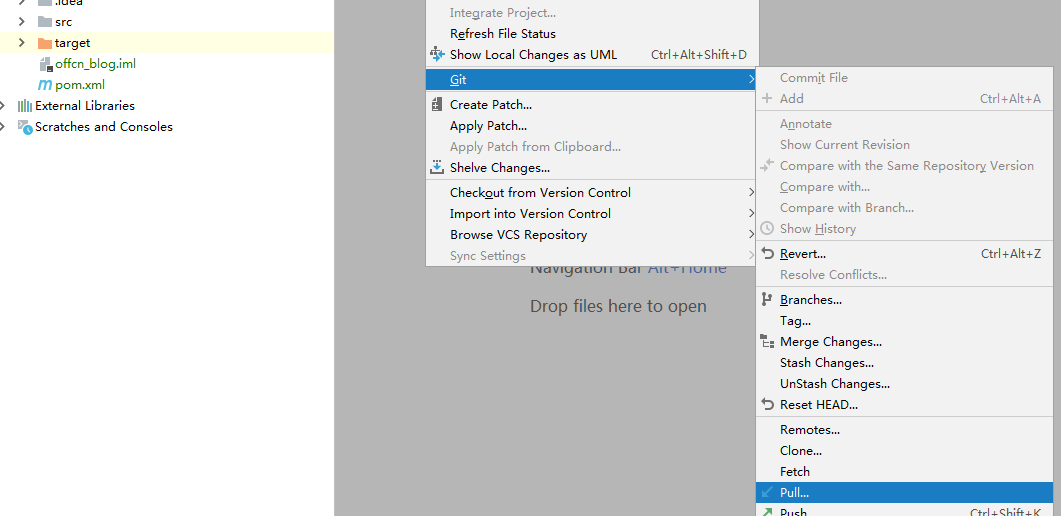


选择菜单栏中的对勾进行commit



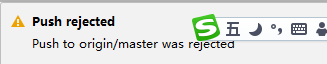


#### 推送到远程仓库



如果出现被拒绝，下面图片错误，在该本地仓库中右键打开git bash here

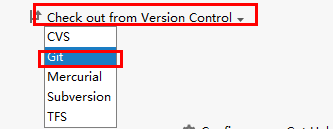
输入git pull origin master --allow-unrelated-histories

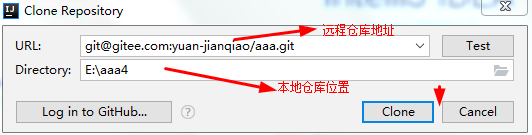


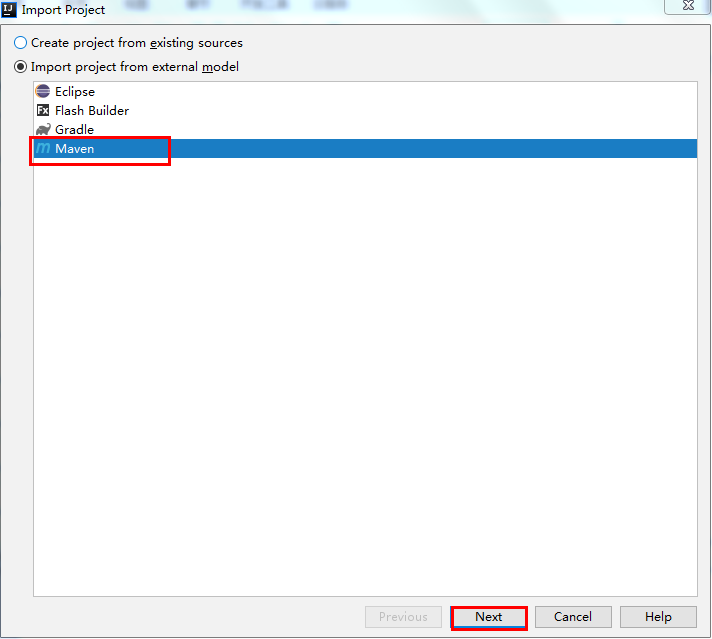
再次选择cvs-->git->push进行推送

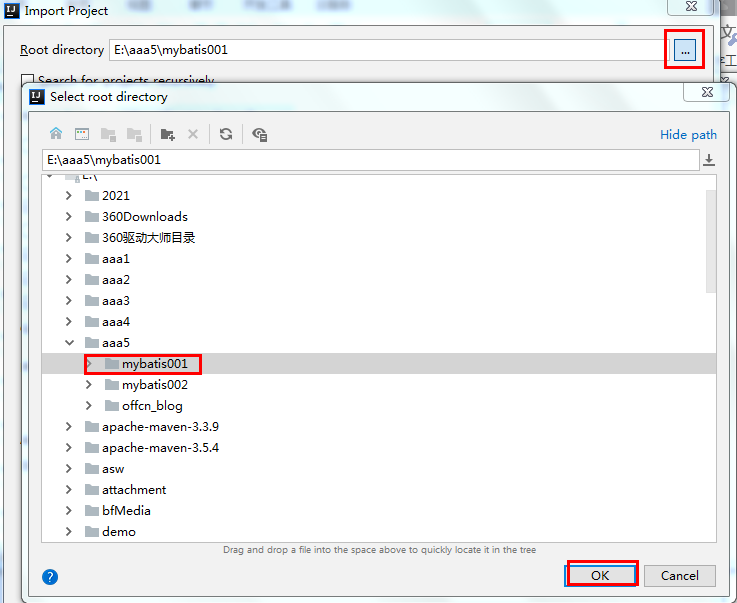
#### 克隆远程仓库到本地

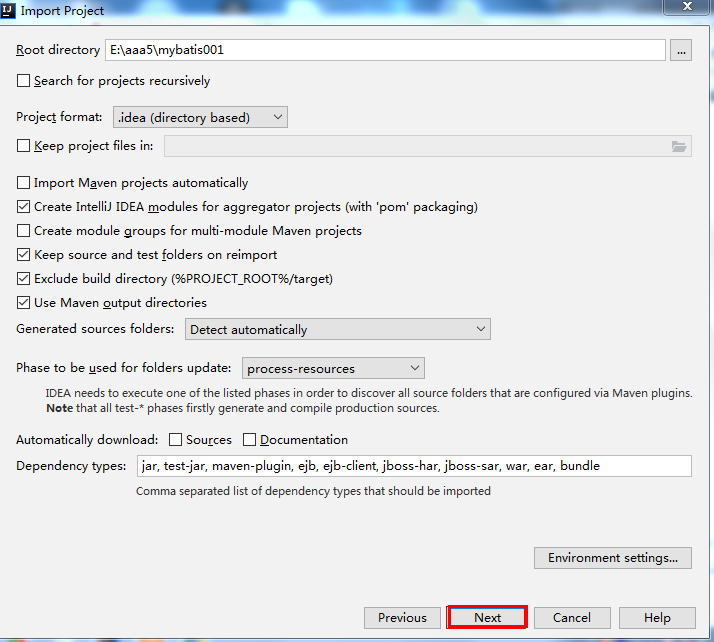
1. 打开idear

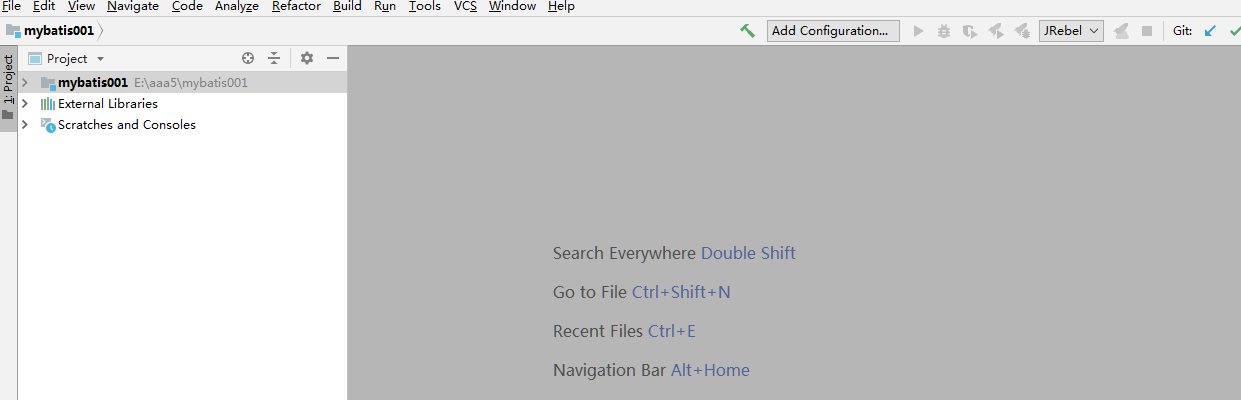




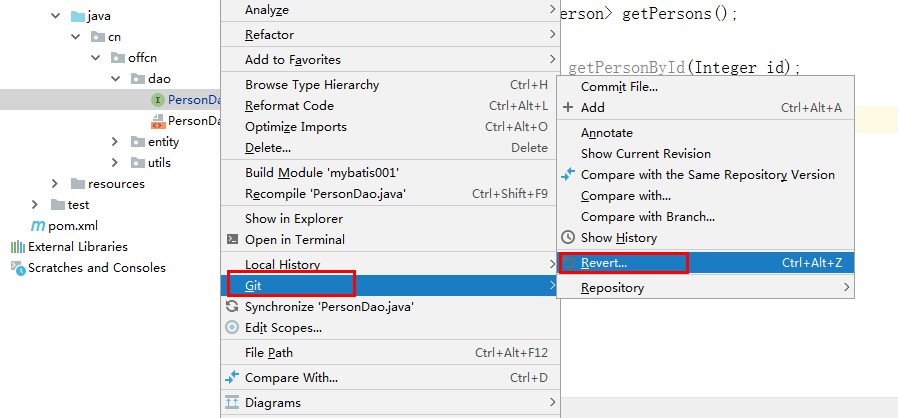


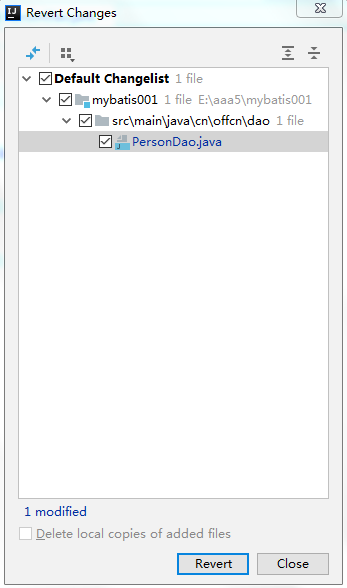






#### 重置文件到修改前

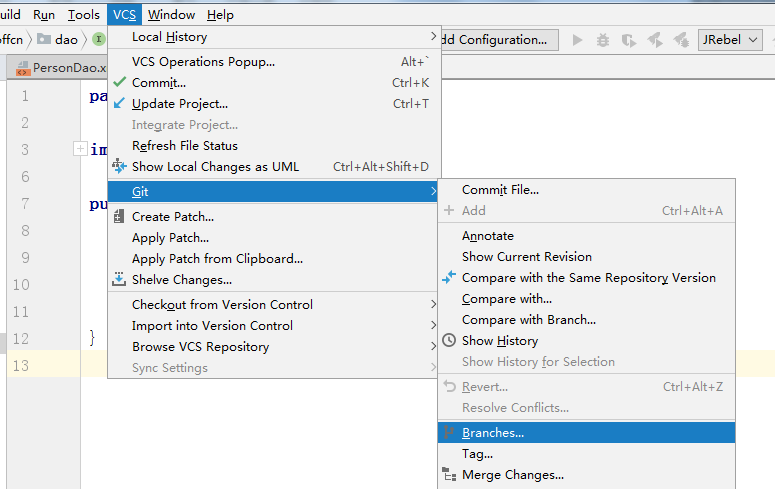


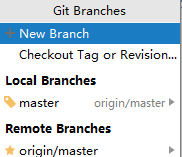


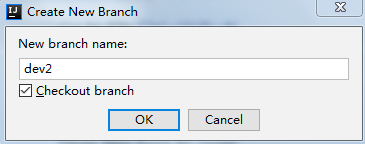
### Idea操作分支

#### 创建分支

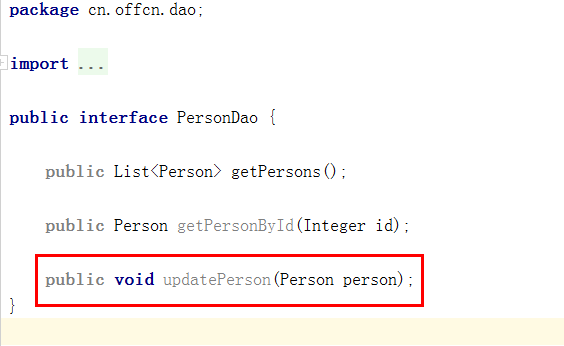
在cvs中选择git -->branchs



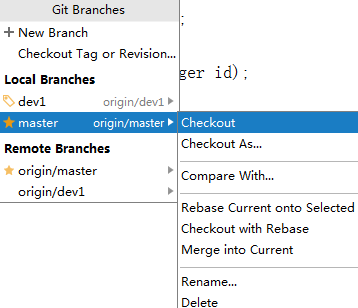


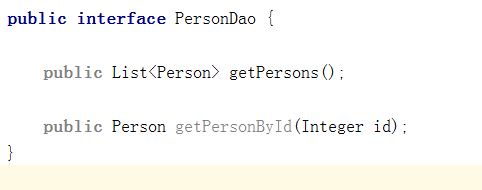


1.在接口中添加一个更新方法

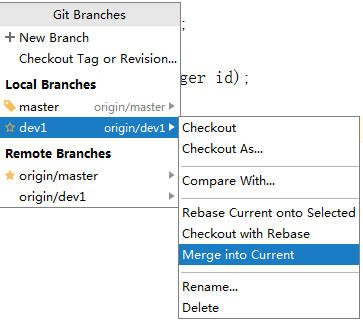


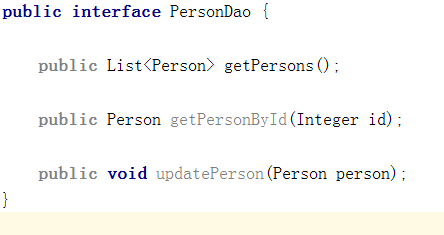
1. 然后添加到暂存区、提交
2. 切换到主分支



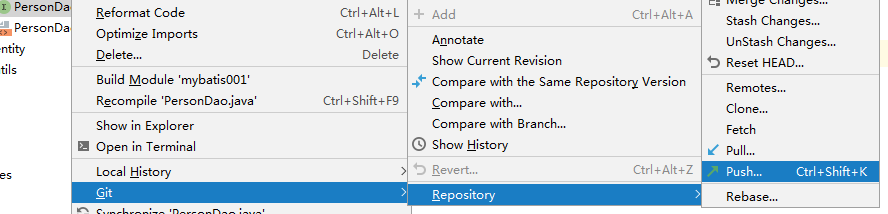


1. 合并到主分支





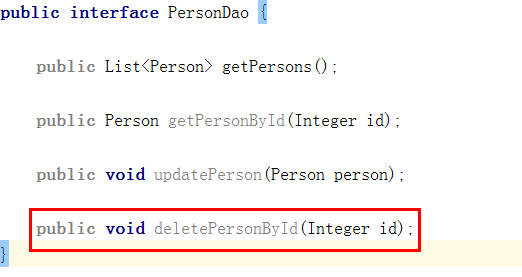
1. 推送到远程仓库



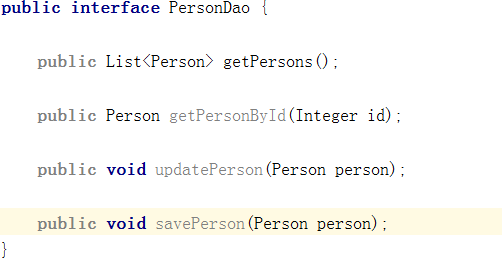


#### Idea解决本地仓库合并问题

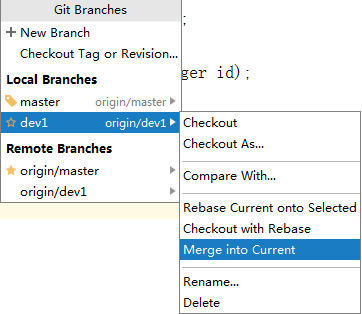
1. 在主分支上对PersonDao添加删除方法并把它添加到本地仓库

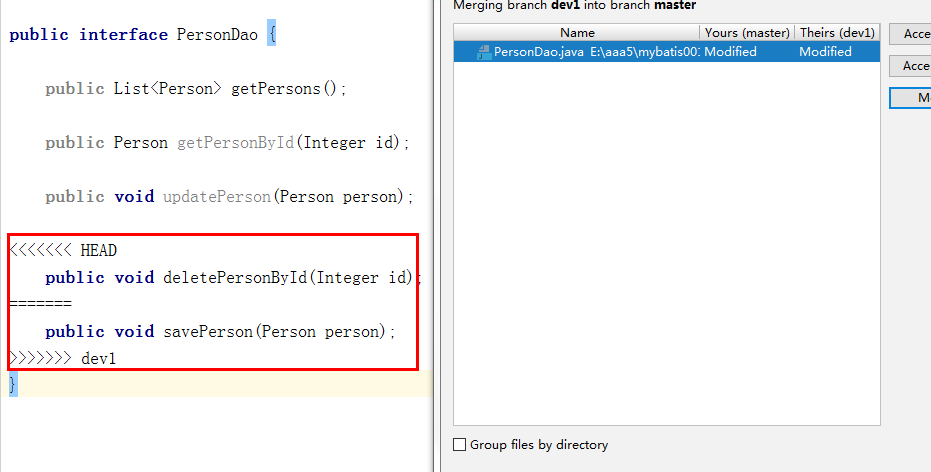


1. 切换到dev1分支，对PersonDao添加保存方法并把它添加到本地仓库

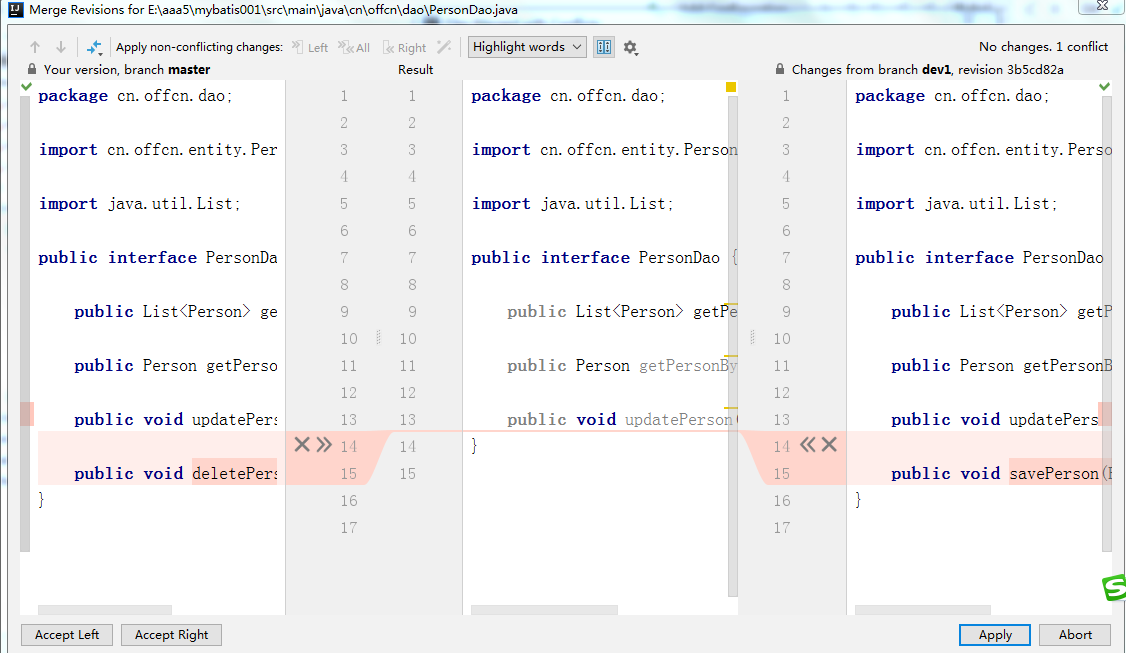


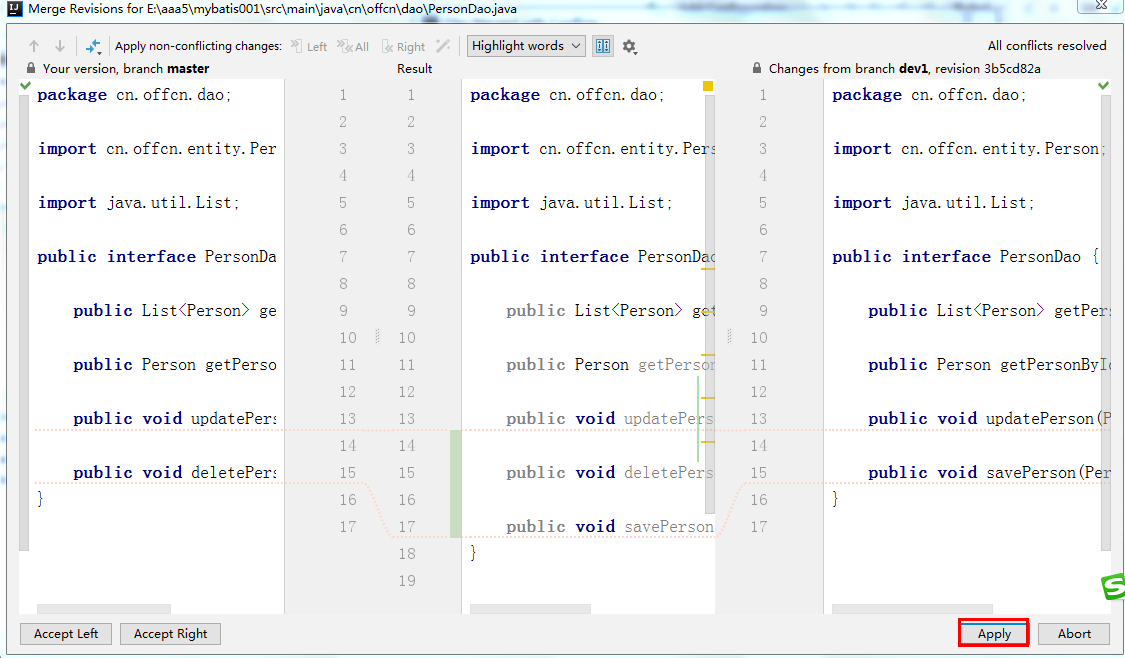
1. 切换回主分支，并把dev1分支合并到主分支



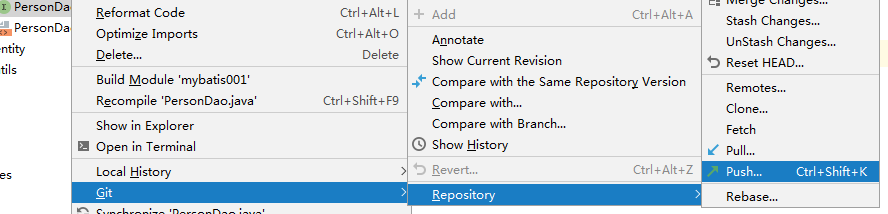


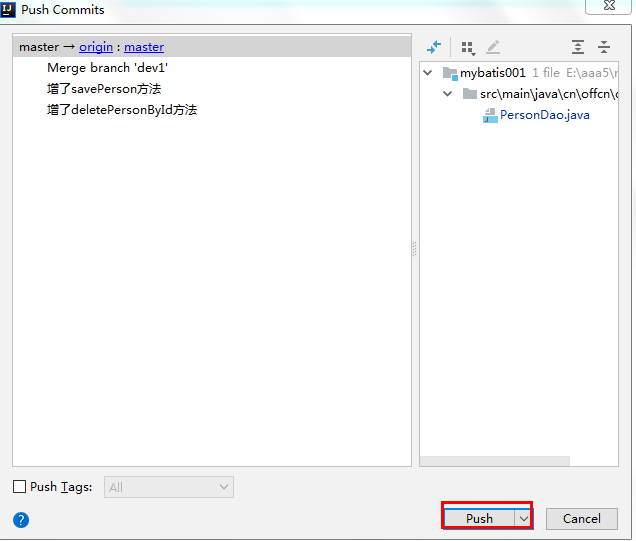
进行合并处理

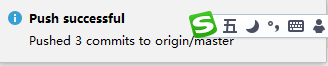




1. 推送

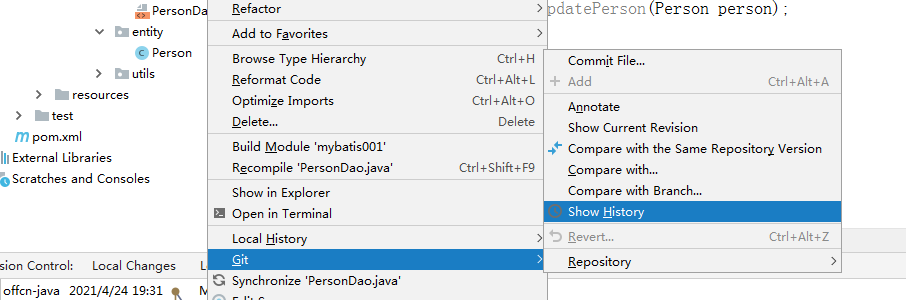




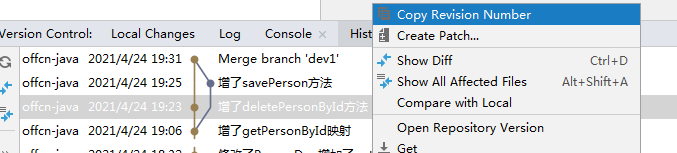


### 回退历史版本

1. 查询历史版本



1. 复制历史版本ID



3.打git-->repository-->Branches..

