Git

尚硅谷Git入门到精通全套教程（涵盖GitHub\Gitee码云\GitLab）_哔哩哔哩bilibili

Git概述

1. 介绍

   Git 是一个免费的、开源的分布式版本控制系统，可以快速高效地处理从小型到大型的各种项目。

   Git 易于学习，占地面积小，性能极快。它具有廉价的本地库，方便的暂存区域和多个工作流分支等特性。

2. 版本控制

   版本控制是一种记录文件内容变化，以便将来查阅特定版本修订情况的系统

   版本控制其实最重要的是可以记录文件修改**历史记录**，从而让用户能够查看历史版本，方便版本切换。

   版本控制工具分为两种：

   （1）集中式版本控制工具

   CVS、SVN（Subversion）、VSS.....

   集中化的版本控制系统都有一个单一的集中管理的服务器，保存所有文件的修订版本，而协同工作的人们都通过客户端连到这台服务器，取出最新的文件或者提交更新。

   缺点：中央服务器的单点故障，无法提交历史记录。

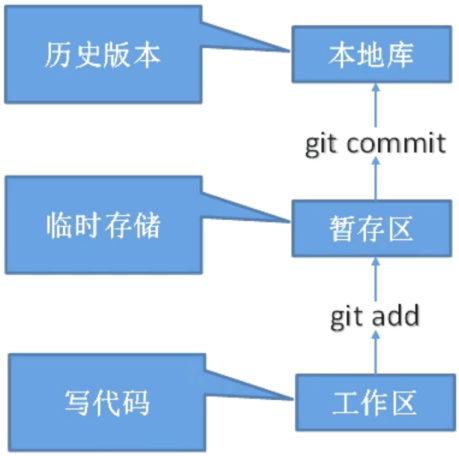
   （2）分布式版本控制工具

   像 Git 这种分布式版本控制工具，客户端提取的不是最新版本的文件快照，而是把代码仓库完整地镜像下来（本地库）。这样任何一处协同工作用的文件发生故障，事后都可以用其他客户端的本地仓库进行恢复。因为每个客户端的每一次文件提取操作，实际上都是一次对整个文件仓库的完整备份。

   优点：

   - 版本控制在本地，可以断网开发
   - 保存完整项目，包含历史记录，更安全

3. 工作机制



- **工作区**写代码，通过`git add`命令添加至**暂存区**，对应本地项目的磁盘空间
- **暂存区**临时存储代码，通过`git commit`提交至**本地库**
- **本地库**记录历史记录，通过`git push`推送至**远程库**

4. 代码托管中心

   代码托管中心是基于网络服务器的远程代码仓库，一般我们简单称为远程库。

Git命令

命令	作用
`git config user.name 用户名`	设置用户签名
`git config user.email 邮箱`	设置用户签名
`git init`	初始化本地库
`git status`	查看本地库状态
`git add 文件名`	添加至暂存区
`git commit -m "日志信息" 文件名`	提交至本地库
`git reflog`	查看历史记录
`git reset --hard 版本号`	版本穿梭

设置用户签名

1. 语法

```
git config --global user.name 用户名
git config --global user.email 邮箱
```

2. 操作

在桌面右键，打开git bash，并输入相应命令。

设置后可以在C盘user文件夹下 .gitconfig 查看。

3. 说明

（1）Git首次安装必须设置用户签名，否则无法提交代码。

（2）这里设置的用户签名和登录github等托管平台的账号**没有关系**。

初始化本地库&查看本地库状态

1. 初始化本地库

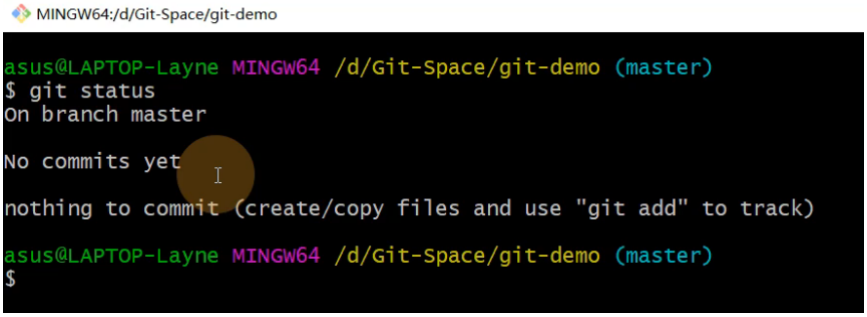
```
git init
```

在项目所在文件夹下，右键打开git bash，输入命令即可。会生成.git文件夹。

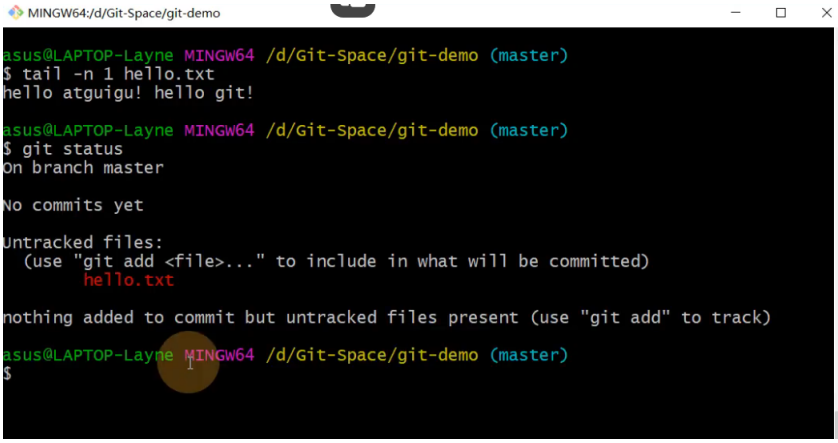
2. 查看本地库状态

```
git status
```

初始化本地库，同一路径下打开bash，使用该命令：



若在当前路径创建了一个文件：



提示我们把文件提交至暂存区。

添加暂存区&提交本地库

1. 添加暂存区

```
git add 文件名
```

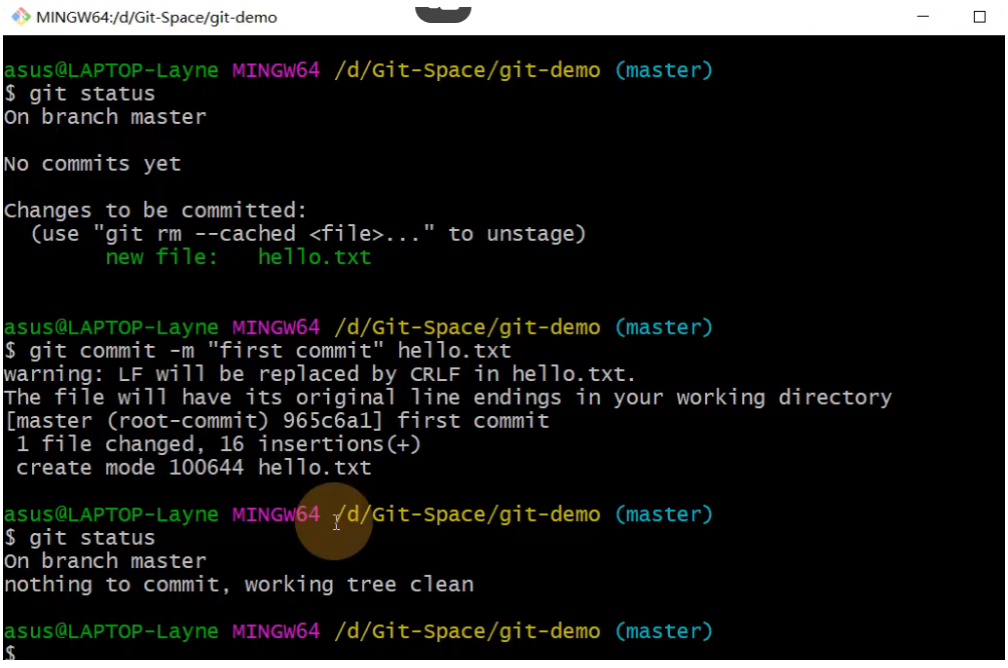
将上面的hello.txt添加暂存区后：

2. 删除暂存区文件

```
git rm --cached 文件名
```

3. 提交本地库

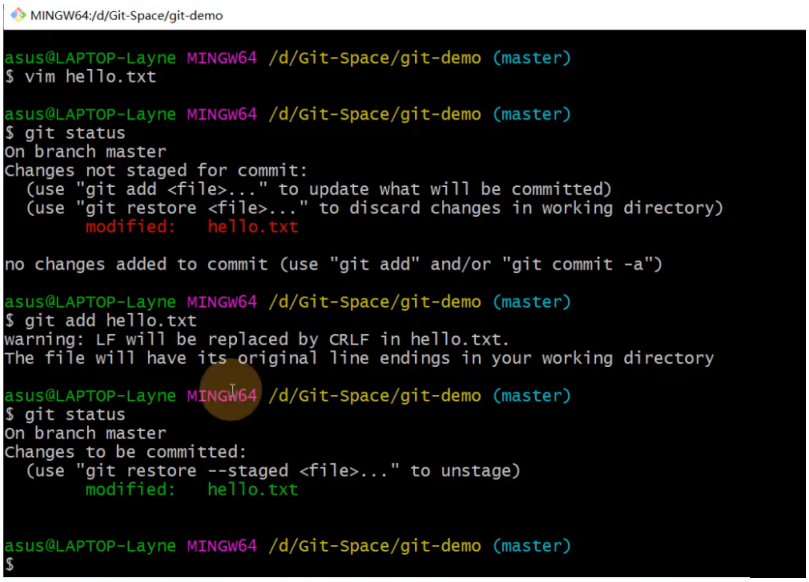
```
# -m 表示添加一个版本日志信息，不写此参数也会打开日志信息的文件框。一般带参数
git commit -m "日志信息" 文件名
```

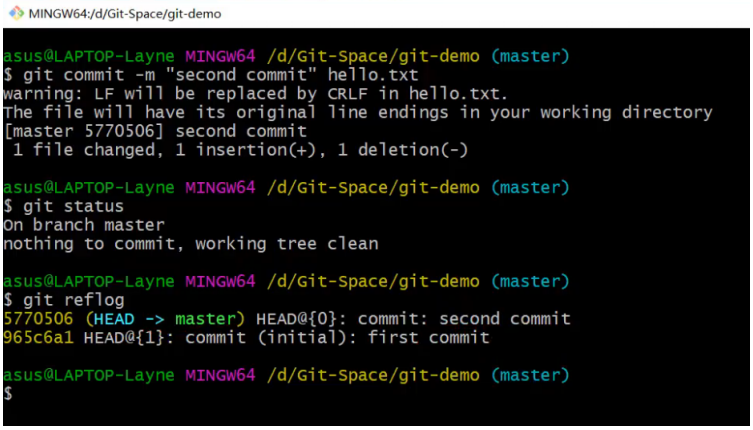


提交后可以查看日志：

修改文件&历史版本

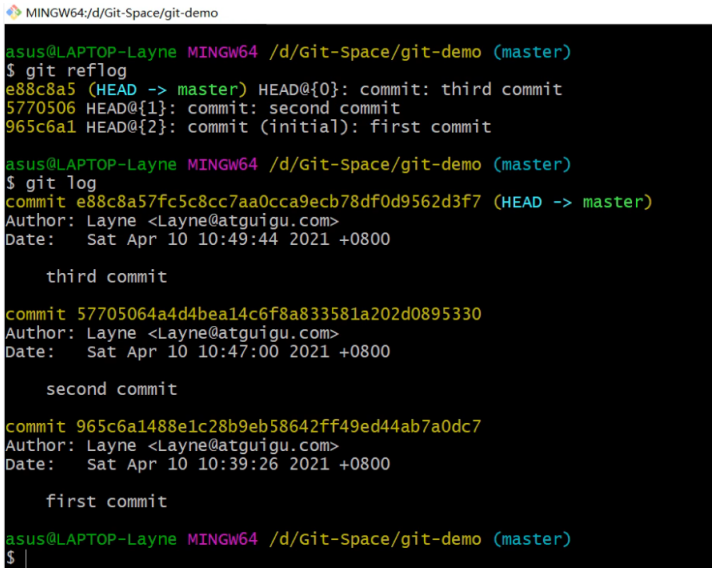
1. 修改文件





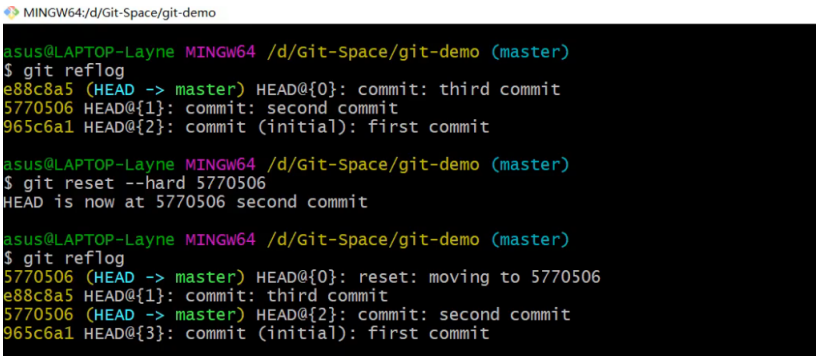
2. 查看历史版本

```
# 查看精简版本信息
git reflog
# 查看详细版本信息
git log
```

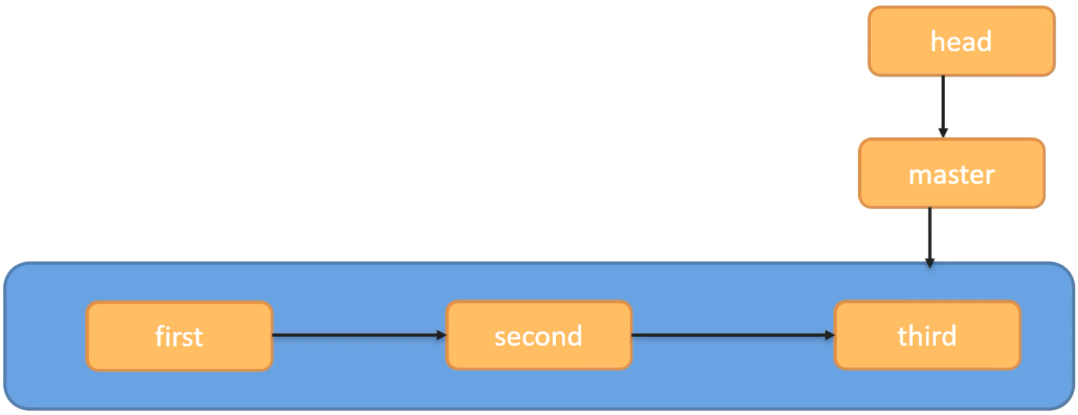


3. 版本穿梭

```
git reset --hard 精简版本号
```



原理：head指针的移动。如下图，head指向分支，分支指向版本。



Git分支操作

分支介绍

1. 什么是分支

在版本控制过程中，同时推进多个任务，为每个任务，我们就可以创建每个任务的单独分支。使用分支意味着程序员可以把自己的工作从开发主线上分离开来，开发自己分支的时候，不会影响主线分支的运行。对于初学者而言，分支可以简单理解为副本，一个分支就是一个单独的副本（分支底层其实也是指针的引用）。

2. 分支的好处

   同时并行推进多个功能开发，**提高开发效率**；

   各个分支在开发过程中，如果某一个分支开发失败，**不会对其他分支有任何影响**。失败的分支删除重新开始即可。

3. 分支操作

命令	作用
`git branch 分支名`	创建分支
`git branch -v`	查看分支
`git checkout` 分支名	切换分支
`git merge` 分支名	把指定的分支合并到当前分支

分支查看/创建/切换

1. 查看分支

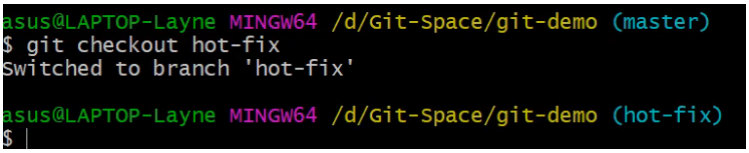
```
git branch -v
```

2. 创建分支

```
git branch 分支名
```

3. 切换分支

```
git checkout 分支名
```

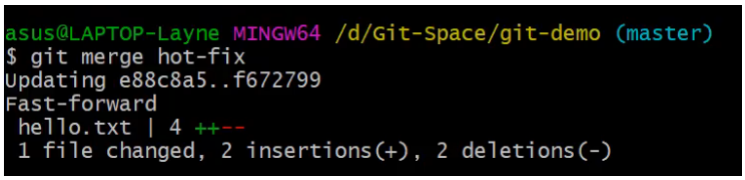


合并分支与冲突

1. 合并分支

   ```
   git merge 分支名
   ```

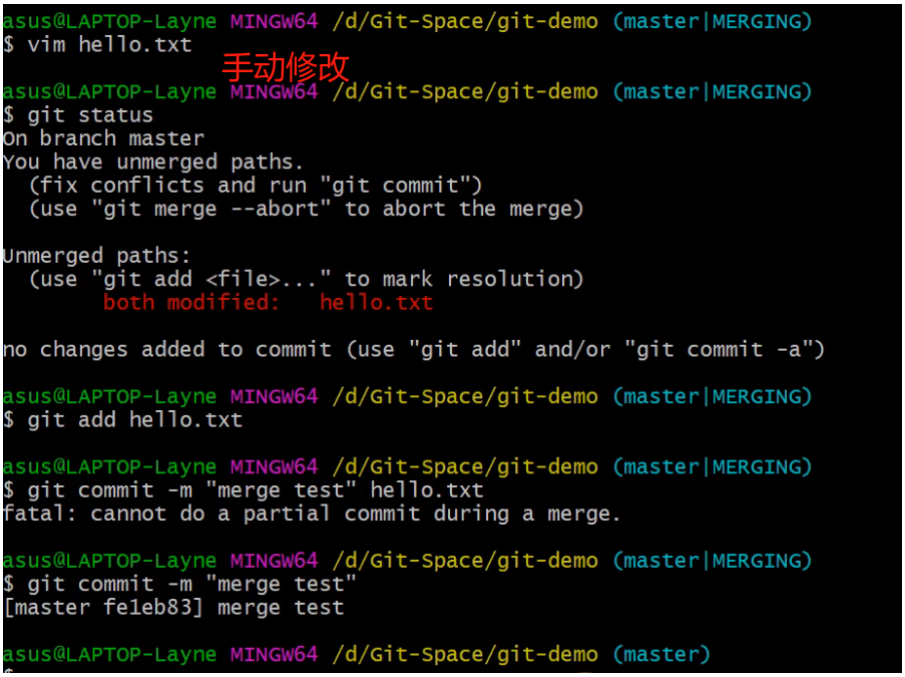
   把“分支名”对应的分支合并到当前分支上。



2. 冲突

   合并分支时，两个分支在同一个文件的同一个位置有两套完全不同的修改。Git无法替我们决定使用哪一个。必须人为决定新代码内容。

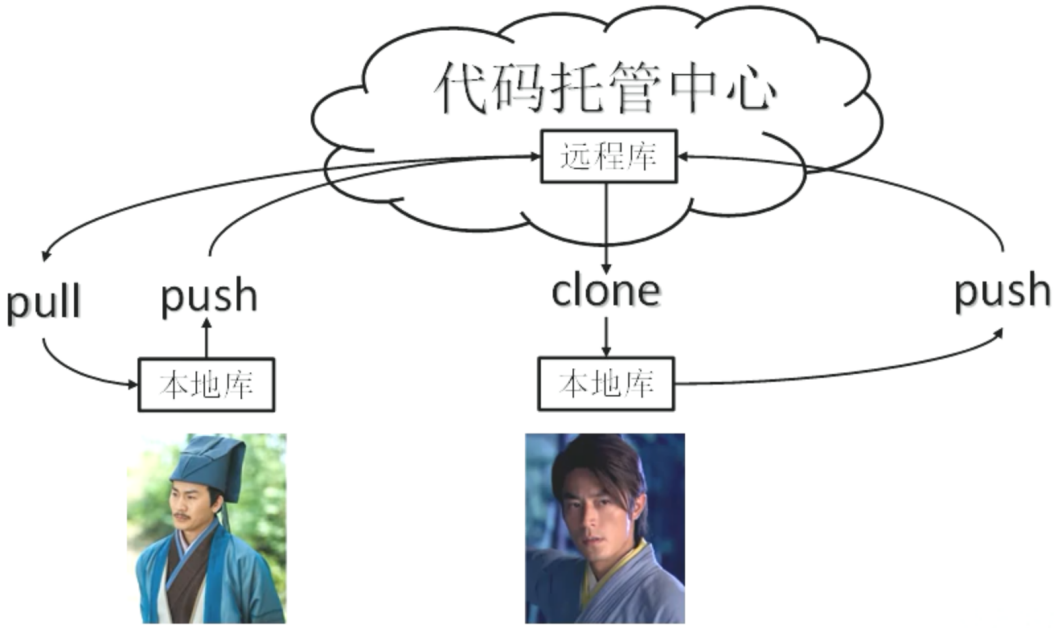
   解决方法：假设当前在master分支中，手动将需要修改的代码确定，然后add和commit。要注意commit时**不能带文件名**。



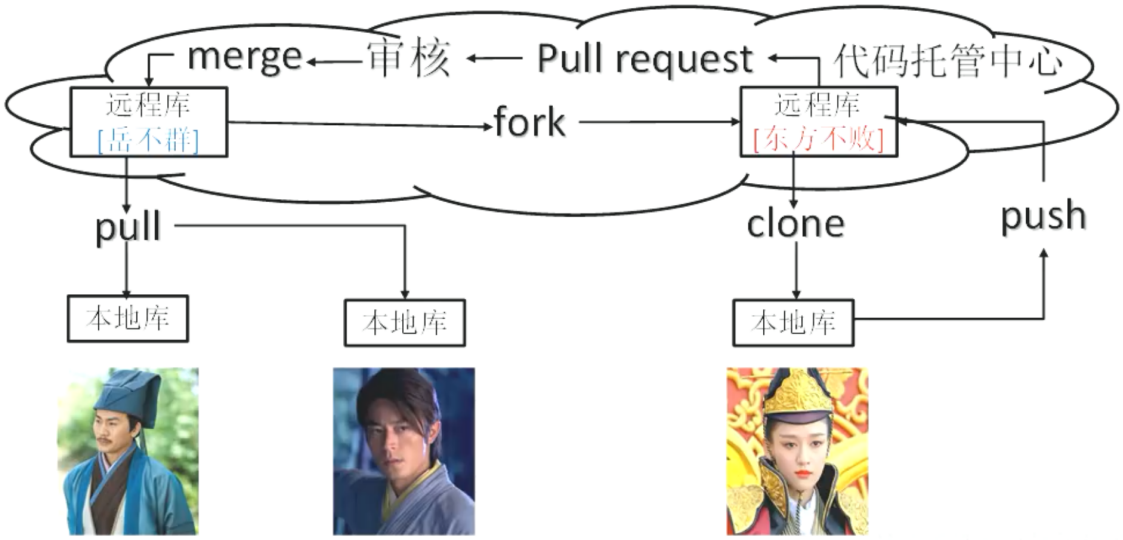
Git团队协作机制

1. 团队内协作

   岳不群把令狐冲加入项目仓库后，令狐冲才能clone代码。



2. 跨团队协作

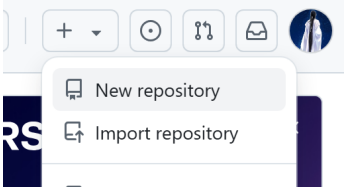


东方不败先将岳不群的远程库fork到自己的远程库，然后clone到本地，修改后可以push；

东方不败可以向岳不群发起pull request请求，岳不群收到后审核，审核通过后将修改merge到自己的远程库。

Github操作

创建远程仓库



Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere?
Import a repository.

Required fields are marked with an asterisk (*).

Owner * Repository name *

Kevin-Xu666 /

Great repository names are short and memorable. Need inspiration? How about redesigned-tribble ?

Description (optional)

Public
Anyone on the internet can see this repository. You choose who can commit.

Private
You choose who can see and commit to this repository.

Initialize this repository with:

Add a README file
This is where you can write a long description for your project. Learn more about READMEs.

Add .gitignore

.gitignore template: None

Choose which files not to track from a list of templates. Learn more about ignoring files.

Choose a license

License: None

A license tells others what they can and can't do with your code. Learn more about licenses.

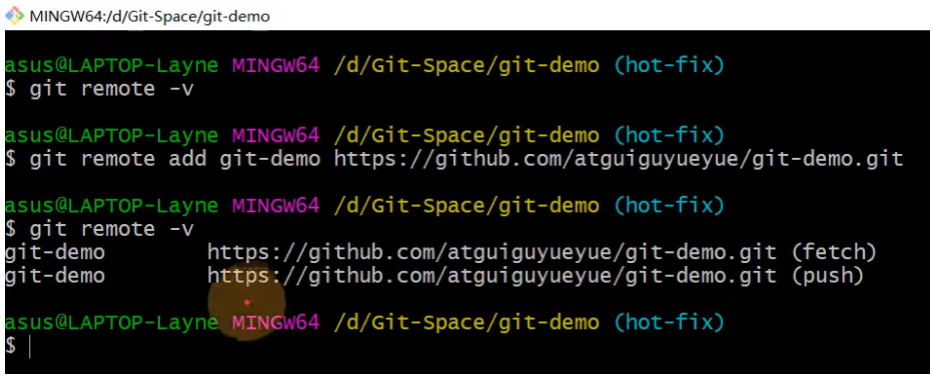
You are creating a public repository in your personal account.

Create repository

创建别名/推送/拉取/克隆

1. 创建远程仓库别名

```
git remote -v # 查看别名
git remote add 别名 远程地址 # 创建别名
```



2. 推送本地库至远程库

```
git push 别名（或远程地址） 分支名
```

3. 拉取远程库到本地

```
git pull 别名（或远程地址） 分支名
```

MINGW64:/d/Git-Space/git-demo

```
asus@LAPTOP-Layne MINGW64 /d/Git-Space/git-demo (master)
$ git pull git-demo master
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 644 bytes | 42.00 KiB/s, done.
From https://github.com/atguiguyueyue/git-demo
 * branch            master     -> FETCH_HEAD
   fe1eb83..63e503d  master     -> git-demo/master
Updating fe1eb83..63e503d
Fast-forward
 hello.txt | 1 +
 1 file changed, 1 insertion(+)

asus@LAPTOP-Layne MINGW64 /d/Git-Space/git-demo (master)
$
```

4. 克隆远程库到本地

   本地没有代码时采用克隆，本地有代码仅仅需要更新时用拉取。

   ```
   git clone 远程地址
   ```

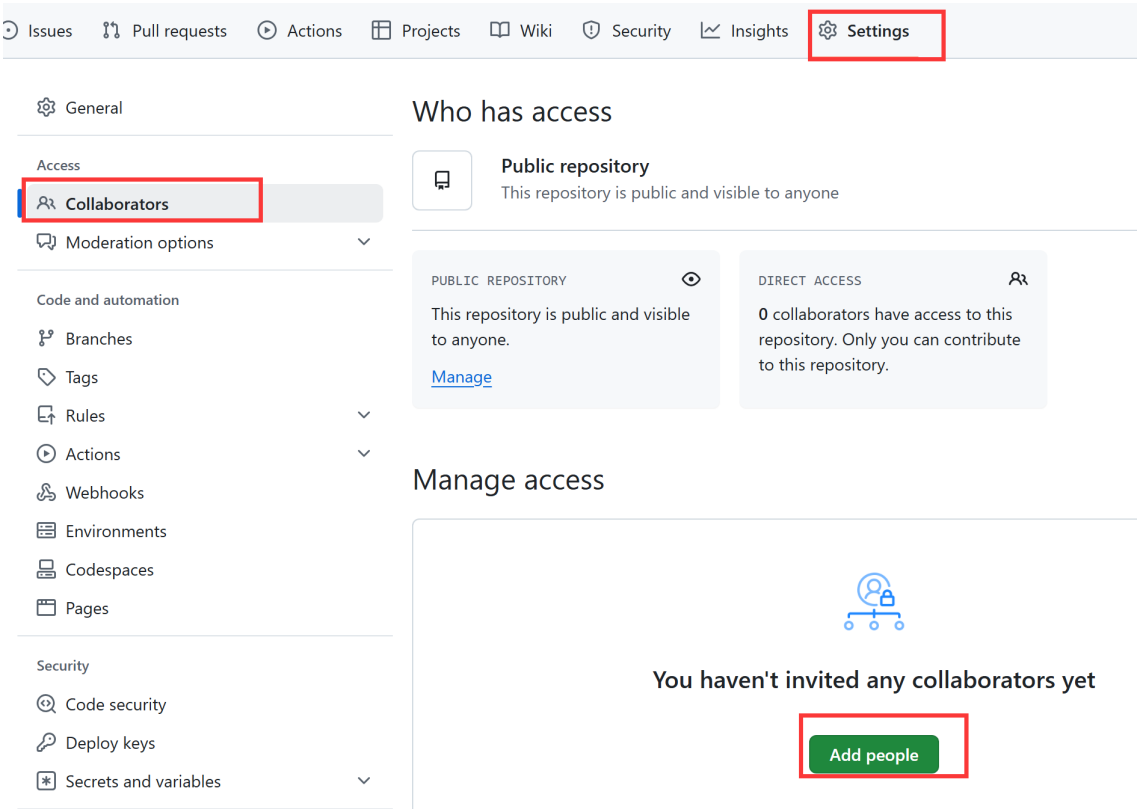
   `clone`命令默认帮我们创建的一个远程仓库名称同名的文件夹，`clone`命令做的事：

   - 拉取代码
   - 初始化本地仓库
   - 创建别名origin

团队协作

1. 团队内协作

   仓库拥有者发起邀请：



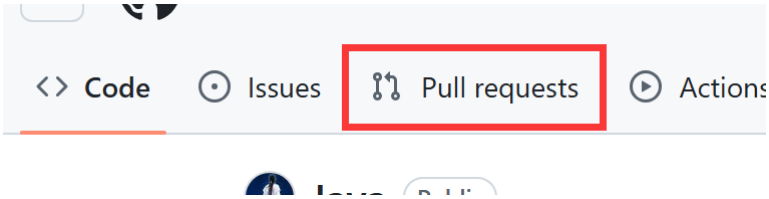
添加后会生成邀请链接，将链接发送给合作者，合作者访问链接选择接收即可。

加入仓库后，可以正常进行推送、拉取、克隆等操作。

2. 跨团队协作

假设团队A要让团队外成员B参与协作，B需要先fork：

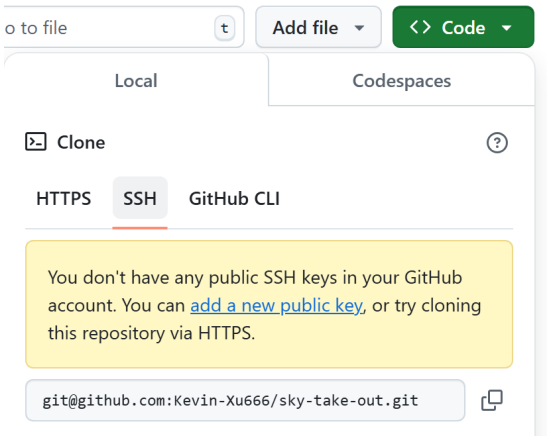
B修改完后，将修改结果pull request：



此时A可以收到该请求，在A的pull request中能对B进行应答，也可以merge新代码。

SSH免密登录

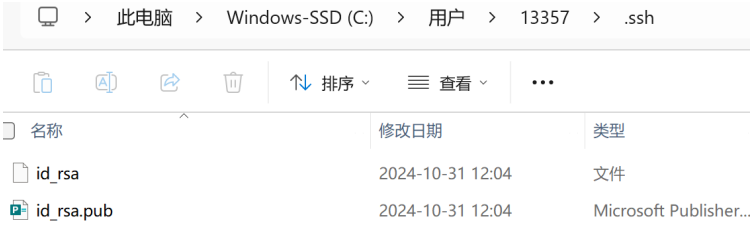
当没有配置时，SSH下显示：



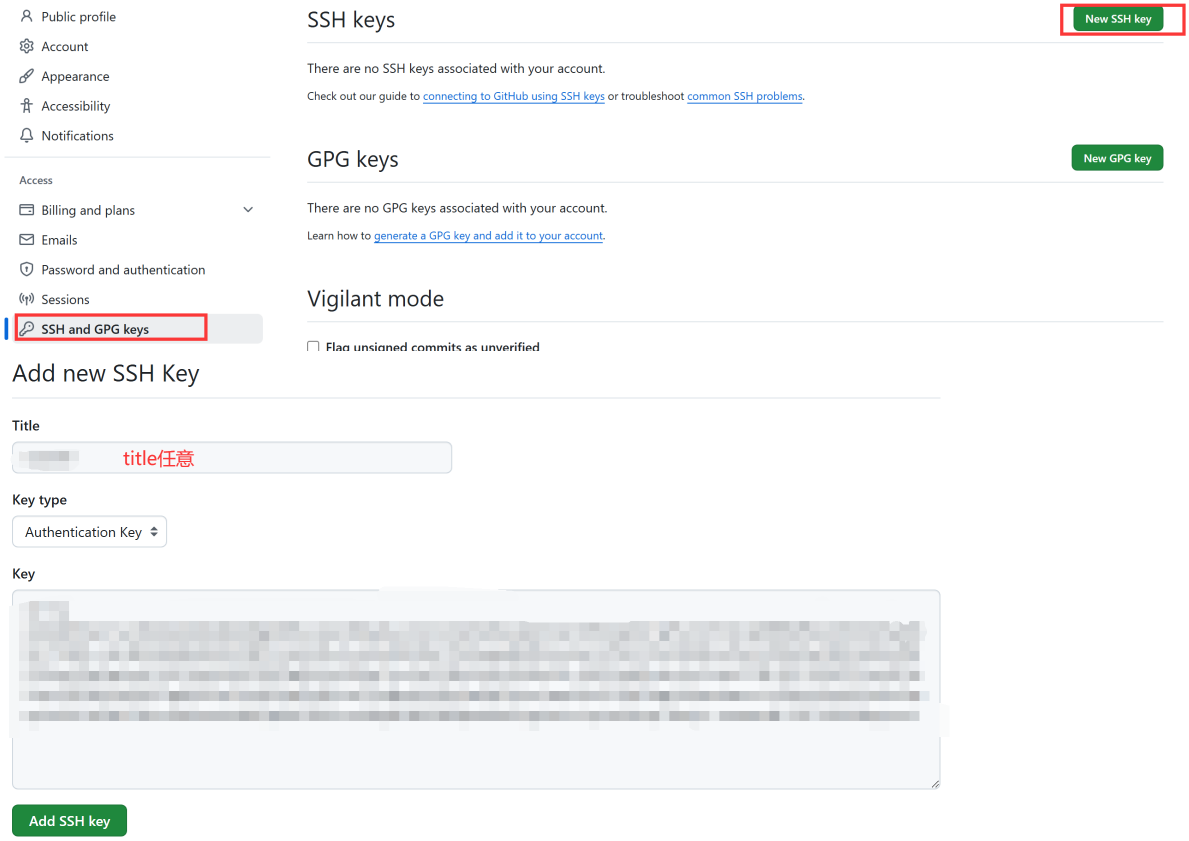
进入C盘用户目录下，右键git bash，输入命令

```
ssh-keygen -t rsa -C 用户@邮箱
```

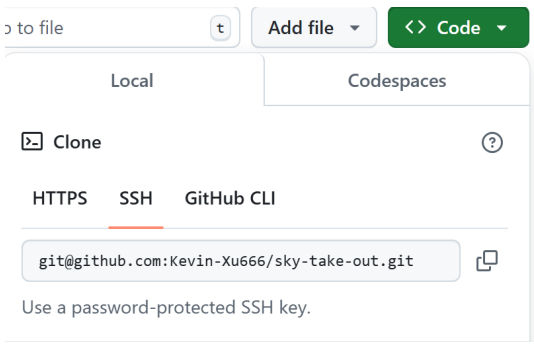
执行后会出现.ssh文件夹，里面有公钥和私钥：



复制公钥内容，将其复制到Github上 头像——>settings——>SSH and GPG keys



设置成功：

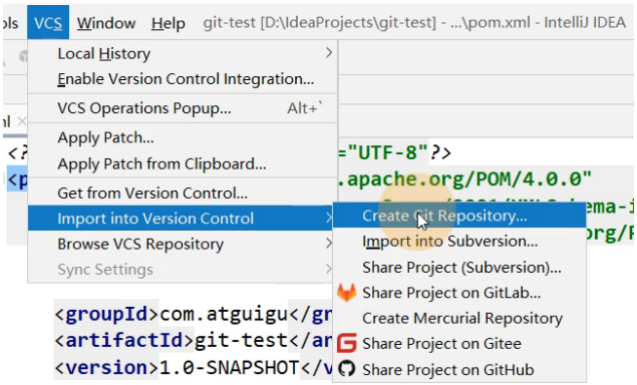


此后可以用SSH方式进行Git

IDEA集成Git

初始化&添加&提交&切换版本

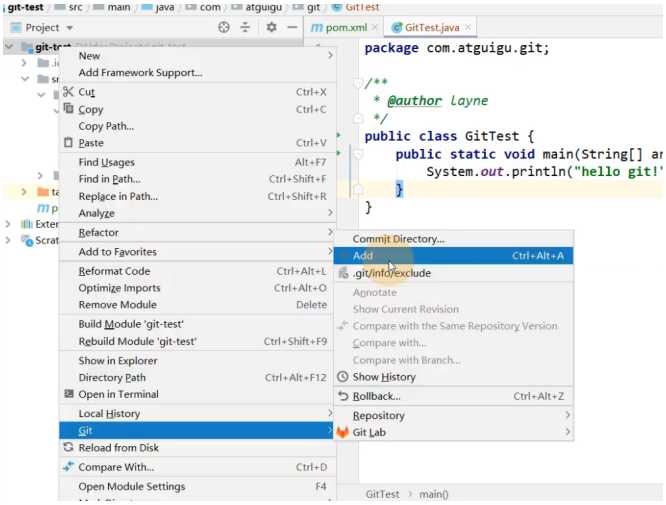
1. 初始化本地库



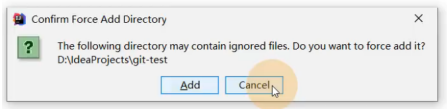
初始化后pom.xml文件名**变红**，说明是未被追踪的状态。

2. 添加至暂存区

选择项目或单个文件进行添加：

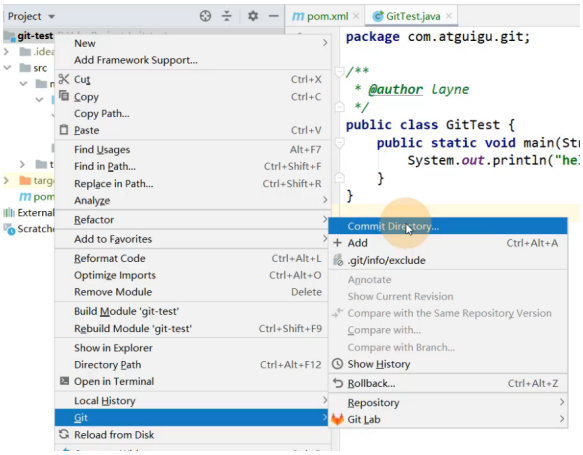


add时要cancel添加ignore文件：



add后文件名会变成**绿色**，代表添加到暂存区，但还没保存至本地库。

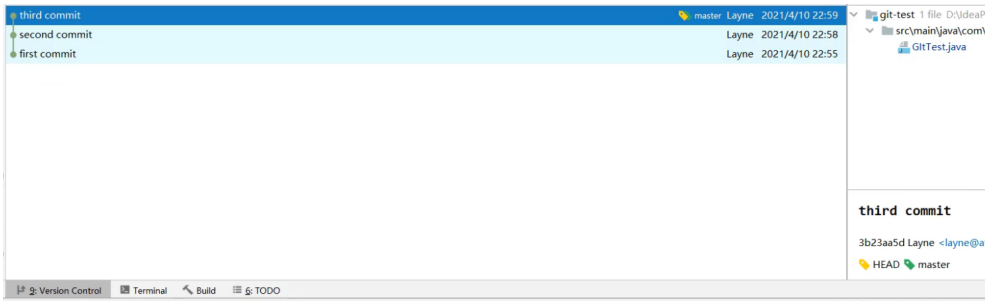
3. 提交至本地库



提交后文件名变成**黑色**。

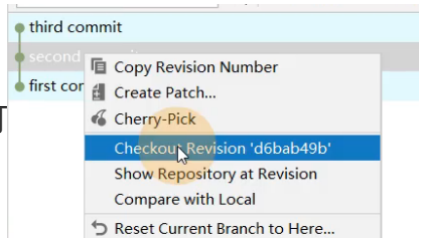
4. 切换版本

先查看版本：



在IDEA最下面Git(Version Control)点击Log，可以查看各版本，同时出现head指针和当前分支指针master的位置。

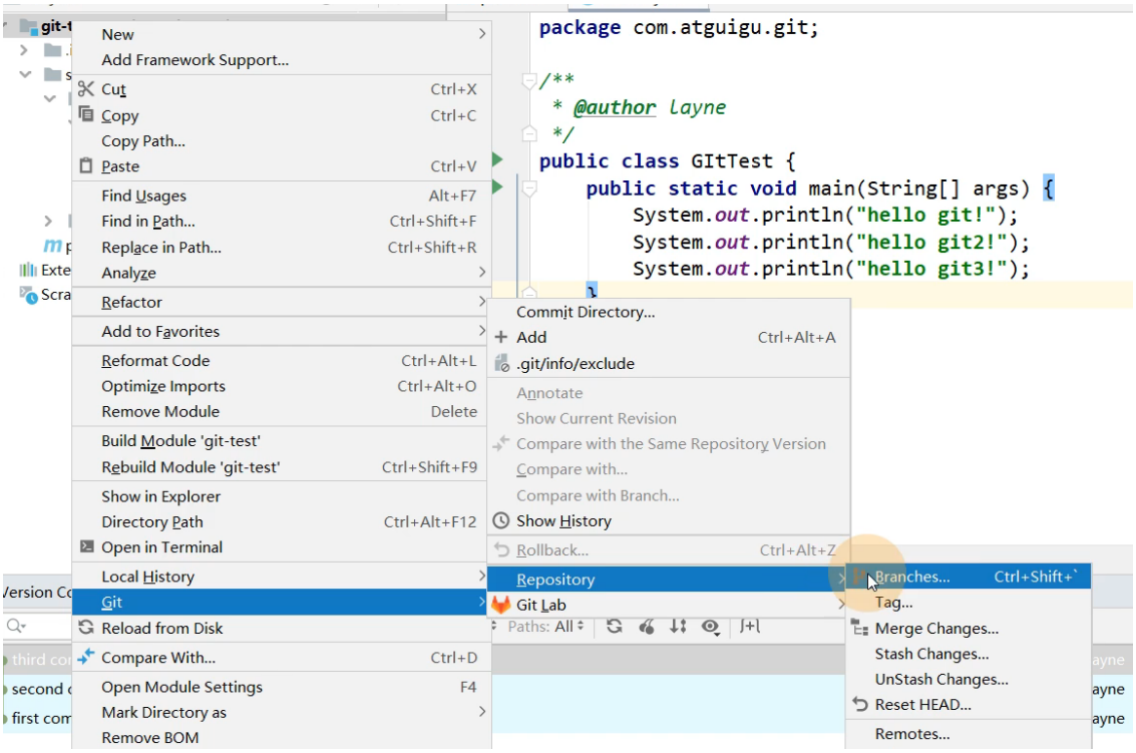
切换版本：选择要切换的版本，右键选择切换即可



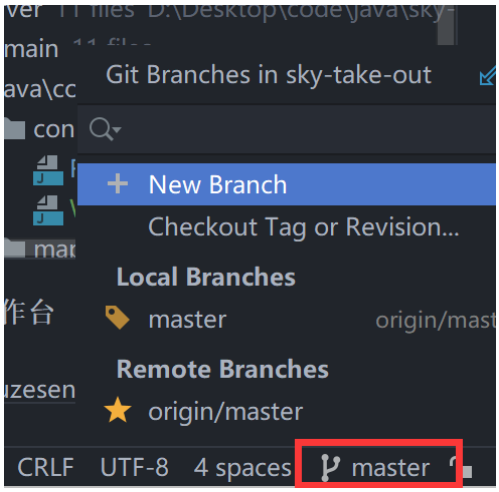
分支创建/切换/合并/冲突合并

1. 创建分支

选择项目并右键：

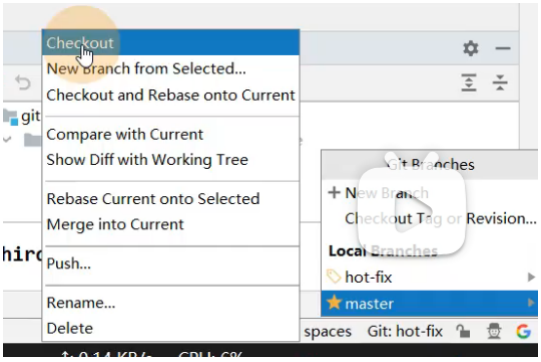


或者直接点击IDEA右下角的分支名：



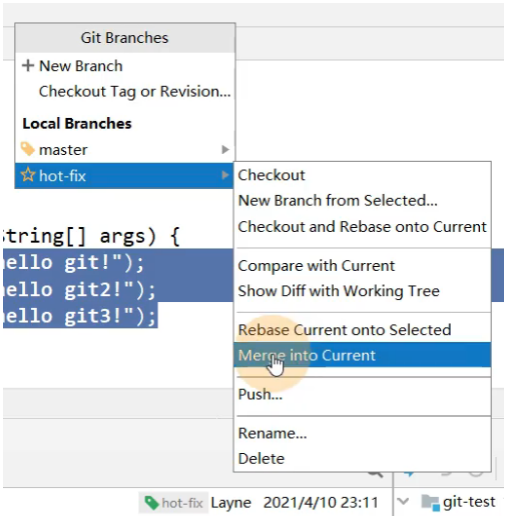
2. 切换分支

点击右下角分支名，选择要切换的分支check out即可。



3. 合并分支（正常）

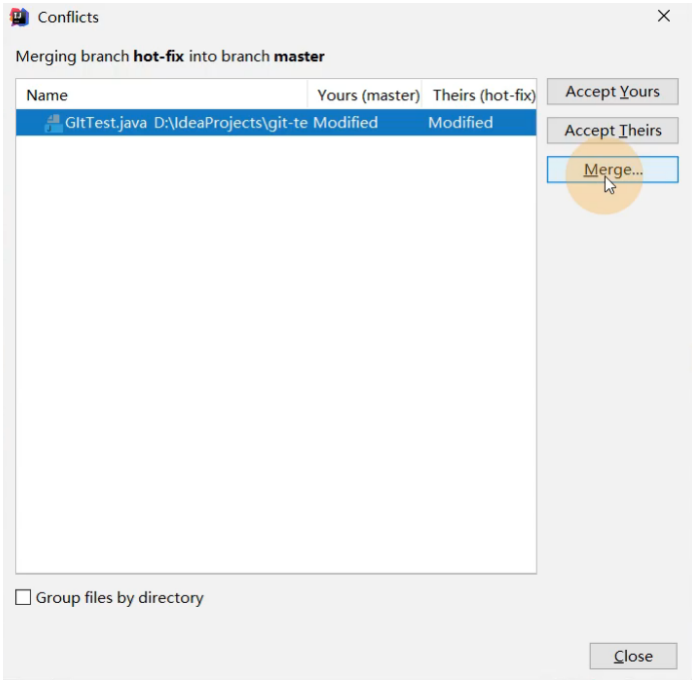
假设想把hot-fix分支合并到master，先进入master分支，IDEA右下角点击分支名，出现对应界面后如下操作：



4. 冲突合并

当hot-fix修改且commit但没有合并到master，此时对master同一文件也做了修改，若想把hot-fix合并到master，会产生冲突。

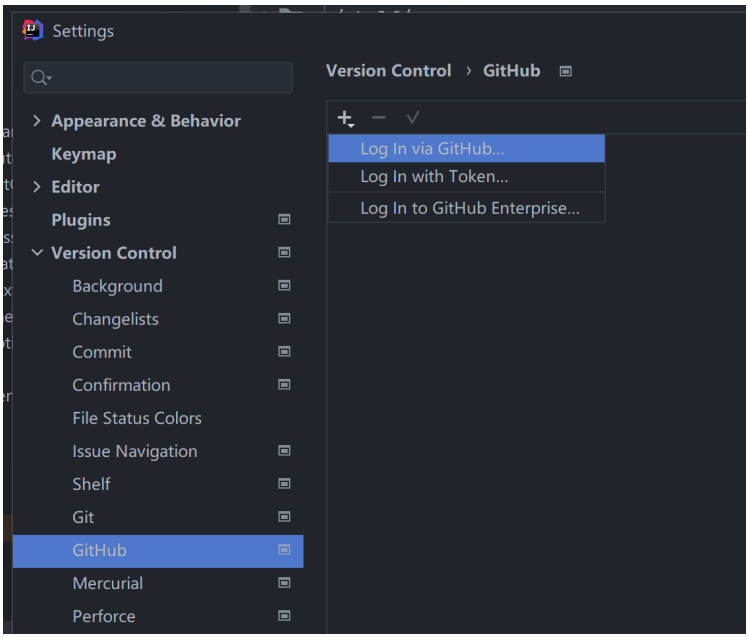
此时按正常的合并分支步骤，出现冲突界面，选择手动merge：



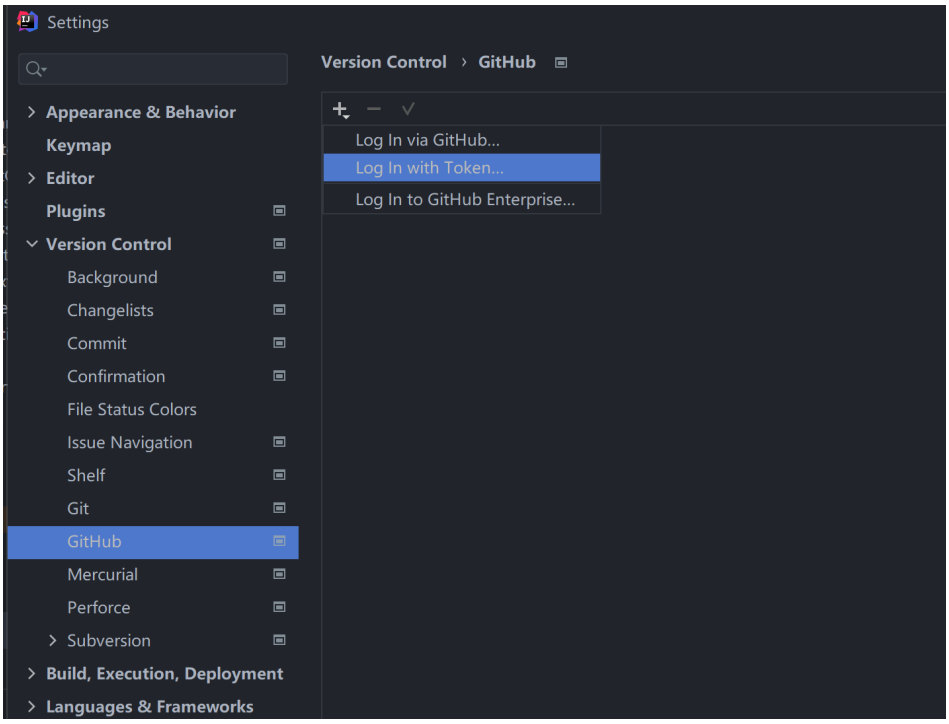
IDEA集成Github

设置Github账号

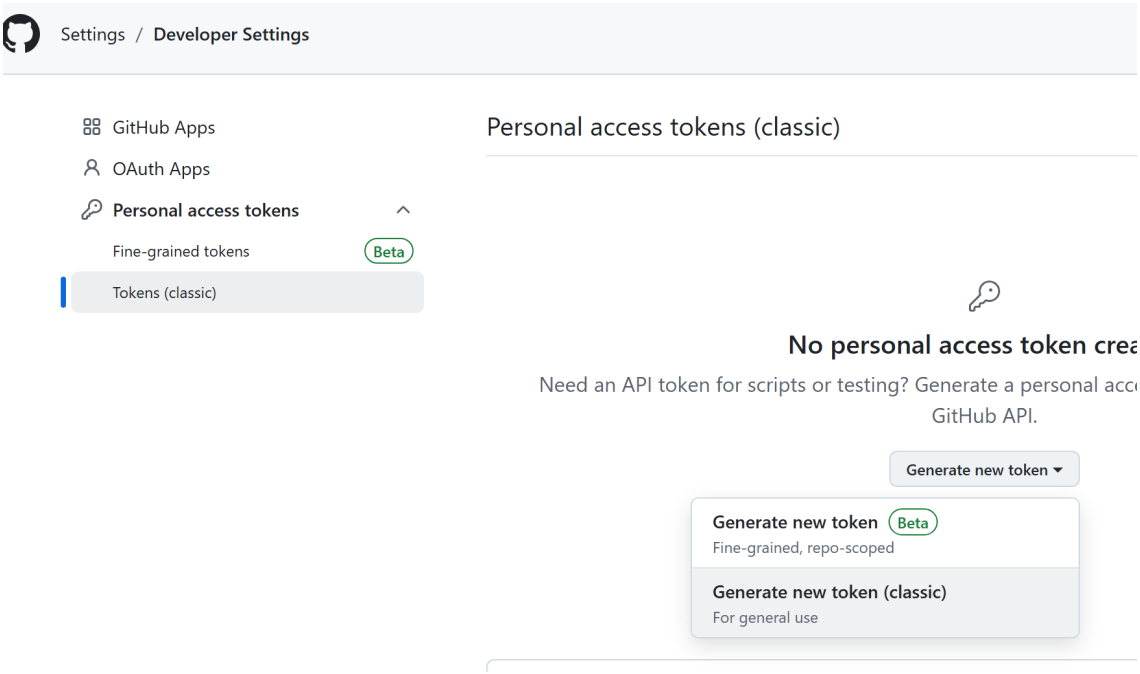
1. 通过账号授权


Settings
Version Control › GitHub
Appearance & Behavior
Keymap
Editor
Plugins
Version Control
Background
Changelists
Commit
Confirmation
File Status Colors
Issue Navigation
Shelf
Git
GitHub
Mercurial
Perforce
Log In via GitHub...
Log In with Token...
Log In to GitHub Enterprise...

2. 通过token


Settings
Version Control › GitHub
Appearance & Behavior
Keymap
Editor
Plugins
Version Control
Background
Changelists
Commit
Confirmation
File Status Colors
Issue Navigation
Shelf
Git
GitHub
Mercurial
Perforce
Subversion
Build, Execution, Deployment
Languages & Frameworks
Log In via GitHub...
Log In with Token...
Log In to GitHub Enterprise...

token生成：

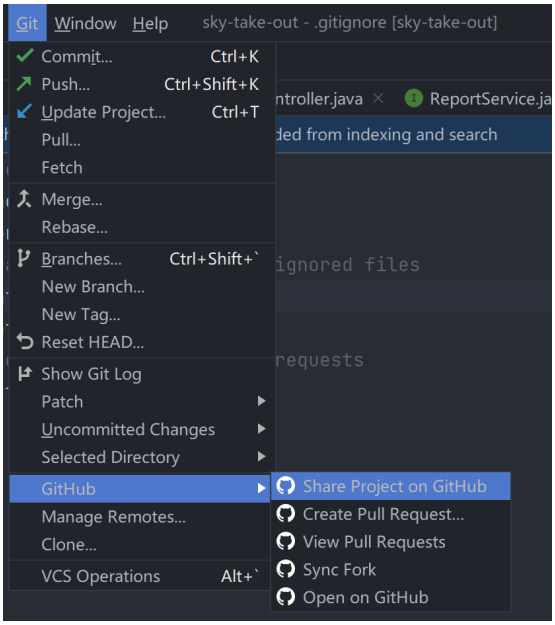


token只会在生成时候**显示一次**，因此要做好保存。

分享工程&Push&Pull&Clone

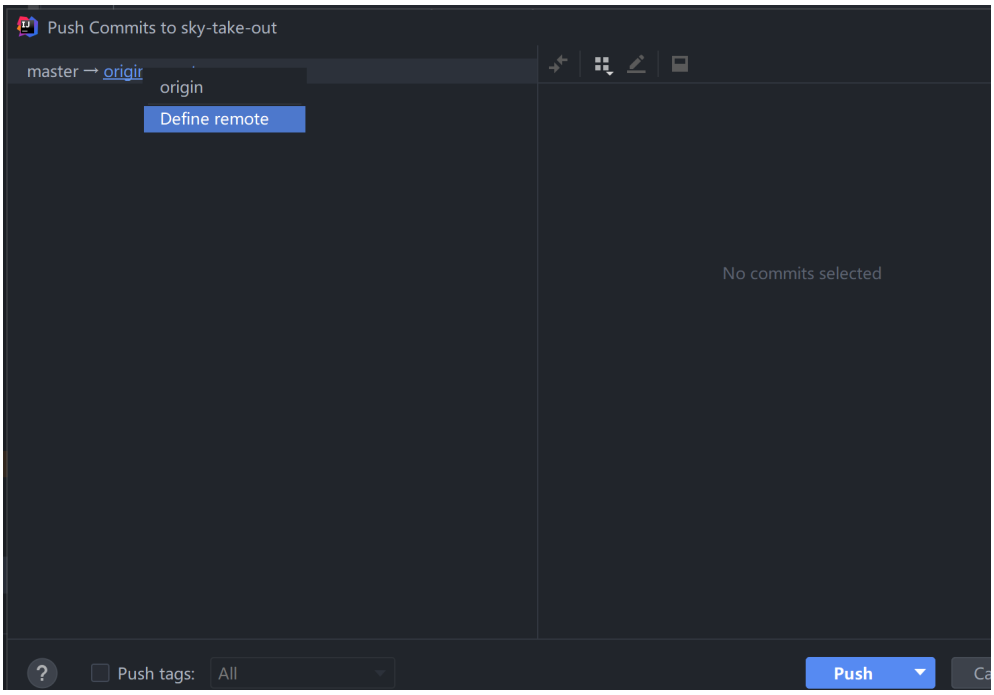
1. 分享工程

   分享工程等价于远程创建项目+push。



2. Push

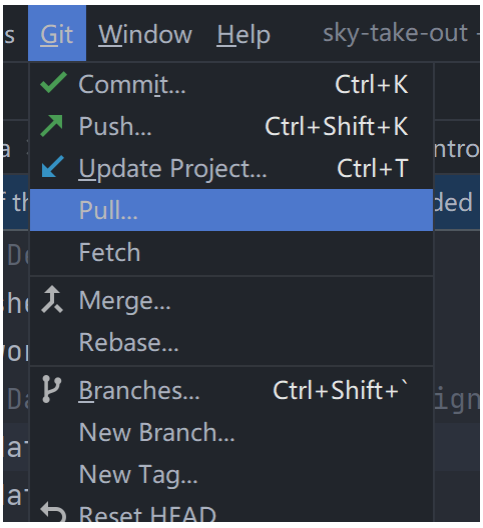
   Git——>Push



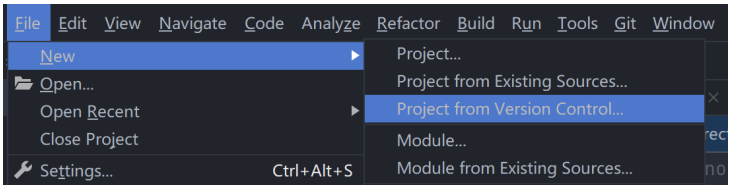
如上图，点击origin，可以自定义远程，在自定义远程输入Github的SSH地址再进行Push，速度会快一些。

3. Pull

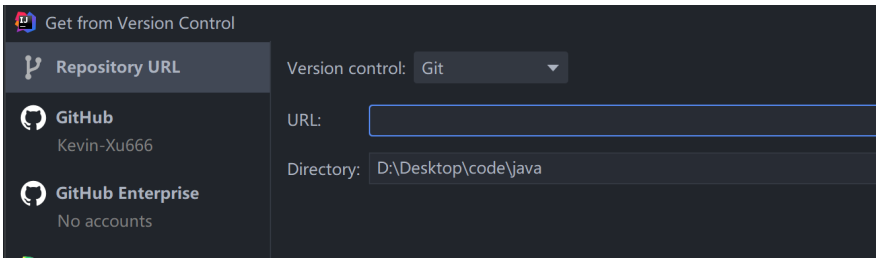
一般来说，要先`pull`拉取一下远程库的代码，将本地代码更新到最新以后，然后再修改，提交，推送！



4. Clone



如果没有打开过文件，初始界面也是选version control。



输入项目链接即可

Gitee

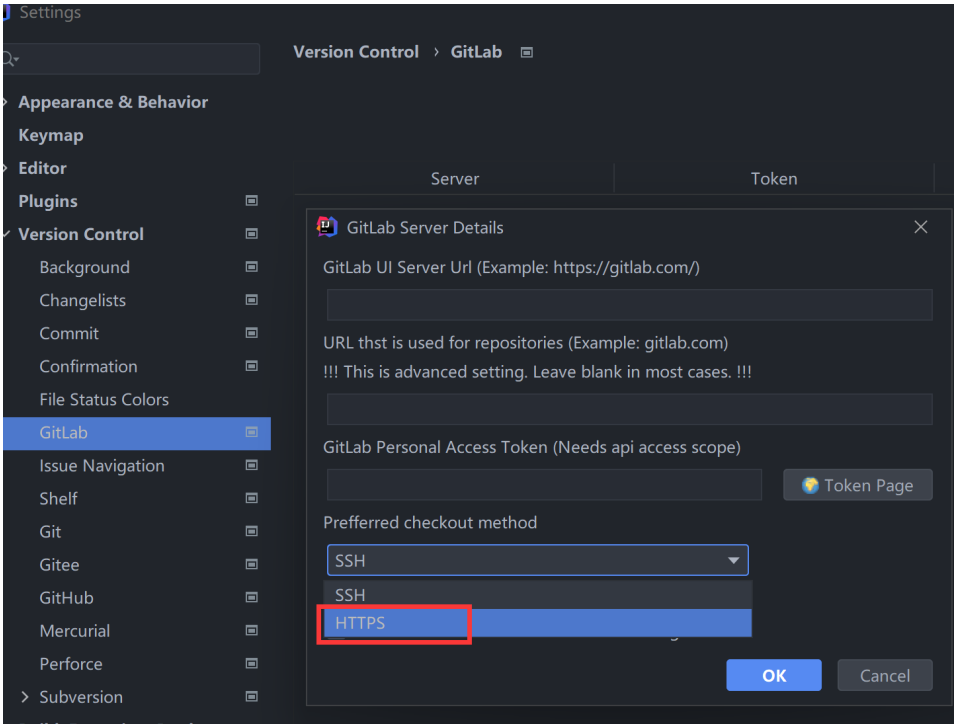
Gitee操作与Github完全相同

1. 导入Github项目

导入后，若Github上项目发生更新，点击刷新即可同步：

GitLab

在IDEA中设置Gitlab时，选择https：



操作上，与Github/Gitee相同。