# IDEA整合Git

# Git概述

## 概述

Git是一个免费、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。其性能优于Subversio、CVS、Perforce和ClearCase等版本控制工具

## 何为版本控制

版本控制是一种记录文件内容变化，以便将来查询特性版本修改情况的系统。

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

## 为什么需要版本控制

个人开发过渡团队协作，如果没有版本控制难以实现代码的版本管理。

## 版本控制工具

### 4.1 集中式版本控制工具

集中版本控制工具像：CVS、SVN（Subversion）、VSS

集中化的版本控制系统诸如CVS、SVN等，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。多年以来，这已成为版本控制系统的标准做法。

这种做法带来了许多好处，每个人都可以在一定程度上看到项目中的其他人正在做些什么。而管理员也可以轻松掌控每个开发者的权限，并且管理一个集中化的版本控制系统，要远比在各个客户端上维护本地数据库来得轻松容易。

事分两面，有好友坏。这么做显而易见的缺点是中央服务器的单点故障。如果服务器宕机一小时，那么在这一小时内，谁都无法提交更新，也就无法协同工作。

### 4.2分布式版本控制工具

分布式控制工具：Git、Mercurial、Bazarr、Darcs等

像Git这种分布式的版本控制工具，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回复。因为每个客户端的每一次文件提取的操作，实际上都是一次对整个文件仓库的完整备份。

分布式的版本控制工具出现之后，解决了几张式版本控制系统的缺陷：

1. 服务器断网的情况也可以就你行开发（因为版本控制是在本地进行的）
2. 每个客户端保存的也都是完整的项目（包含历史记录，更加安全）

## Git简史

1、1991年linus开发linux开源的版本

2、1991-2002年很多开发者贡献了linux系统优化的版本，这个时候基本都是通过linus手动的方式进行代码合并

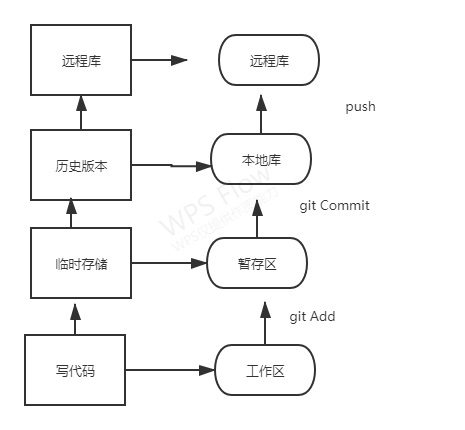
3、2002-2005年BiMover公司处于人道主义精神，商业版本控制软件BitKeeper版本控制系统，授权linux社区免费使用，进项版本管理；但是要求不能进行破解

4、2005年Samba的作者Andrew破解BitKeeper版本控制软件，结果BiMover公司收回了使用权

5、2005年linus自己动手用C语言开发一个分布式控制系统：git ；主题程序开发完成只用了两周，一个月后linux系统代码由Git托管

6/2008年GitHub上线，JQuery，PHP等将代码托管到Git上

## Git工作机制



## Git和代码托管中心

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。

### 7.1局域网

Gitlab

### 7.2互联网

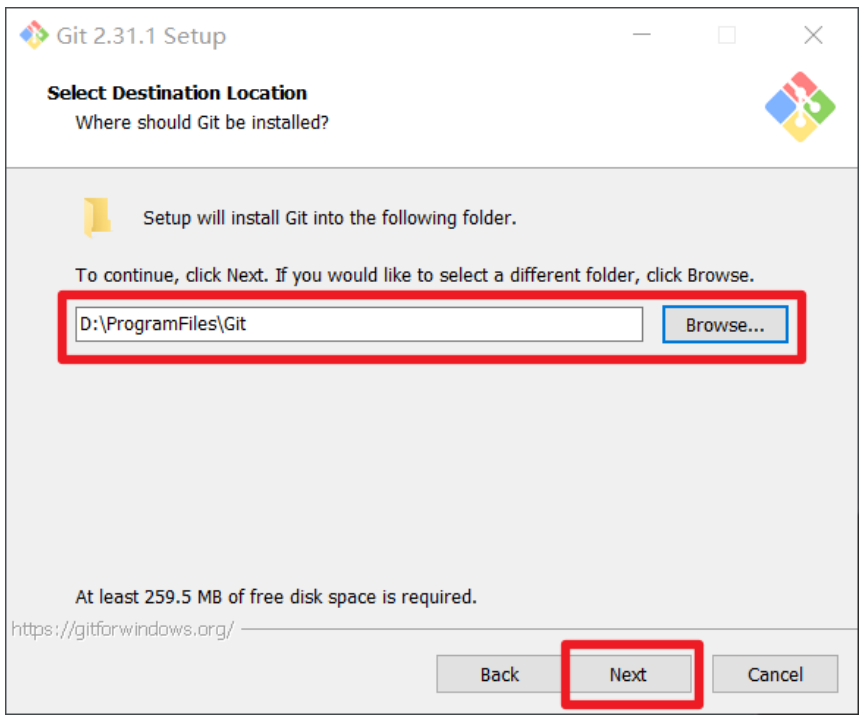
1. GitHub（外网）
2. Gitee 码云（国内网站）

# Git安装

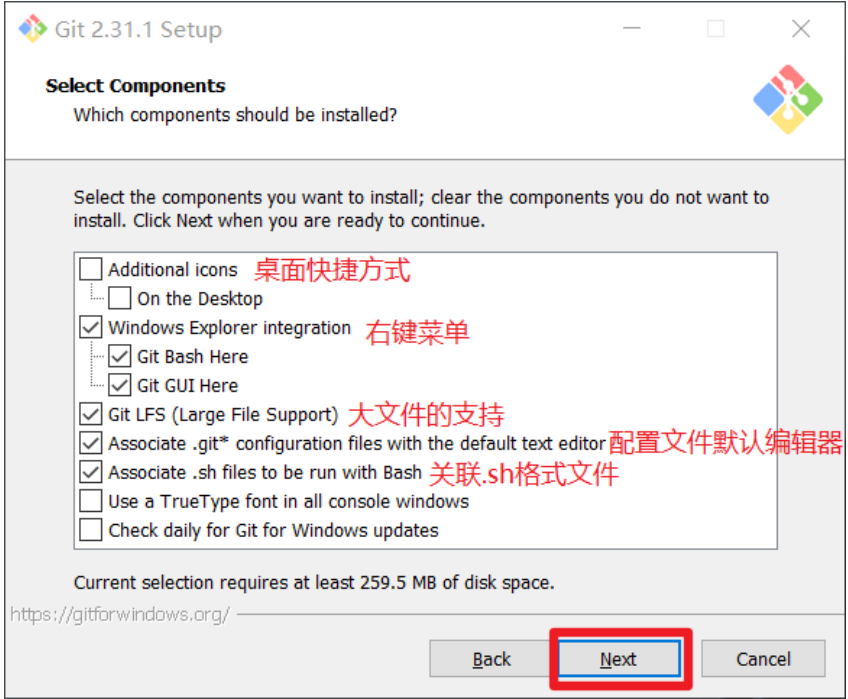
1. 选择安装文件32位或者64位，双击安装文件



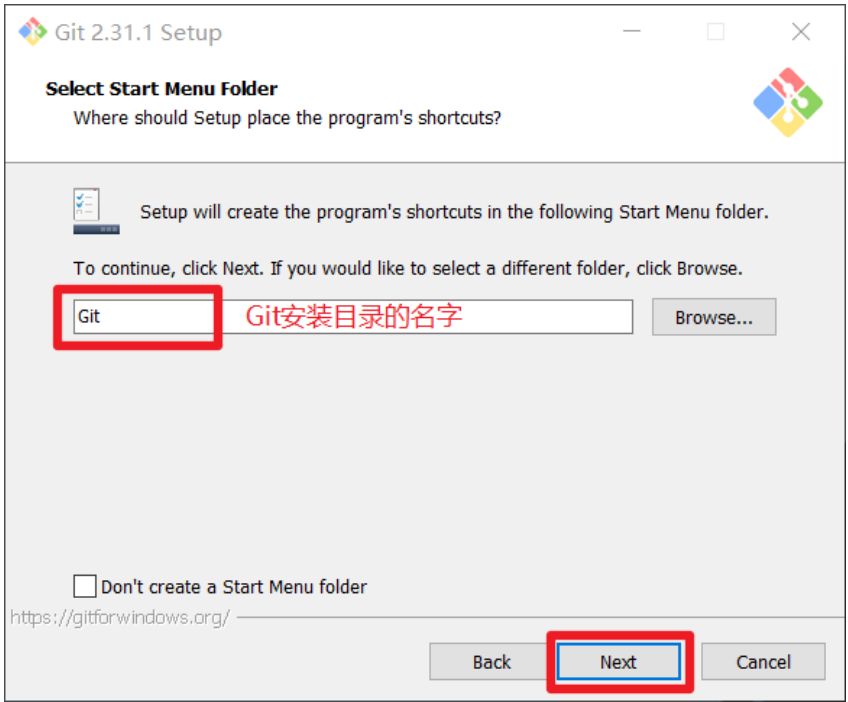
1. 选择Git的安装目录，要求非中文并且没有空格，点击下一步



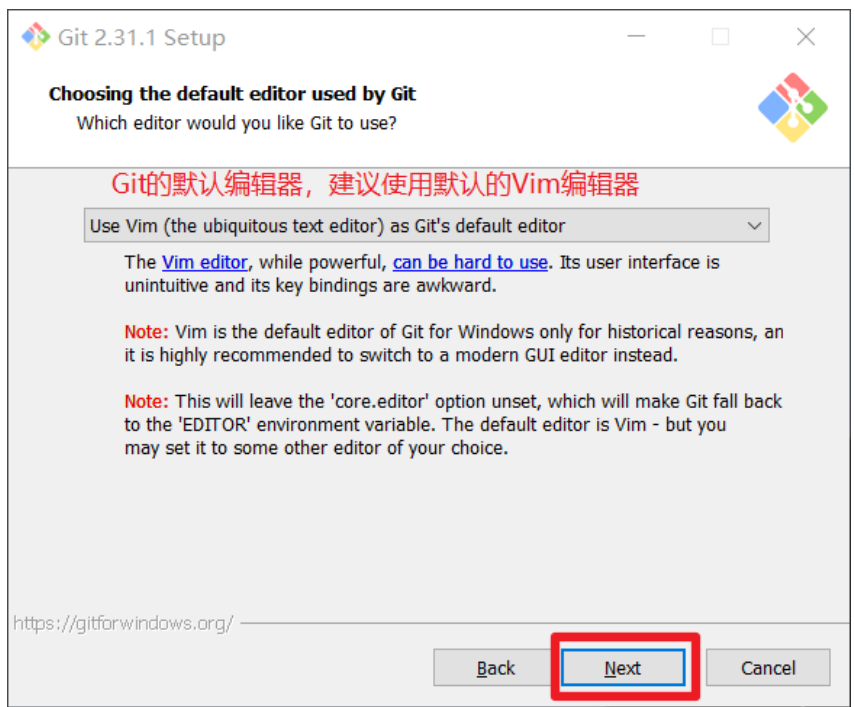
1. Git的选项设置推荐使用默认的，点击下一步



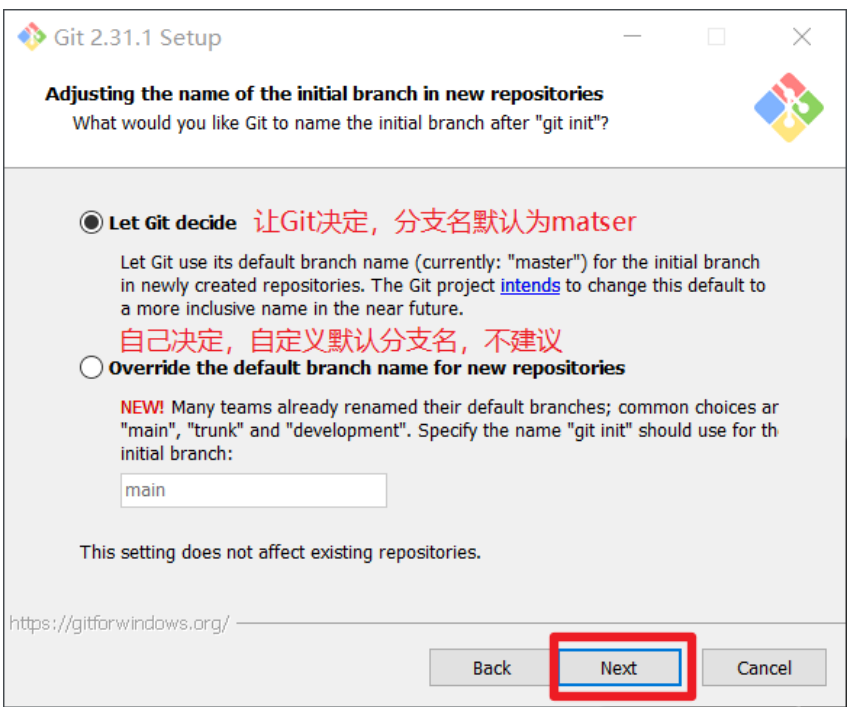
1. Git的安装目录名，不用修改直接点击下一步



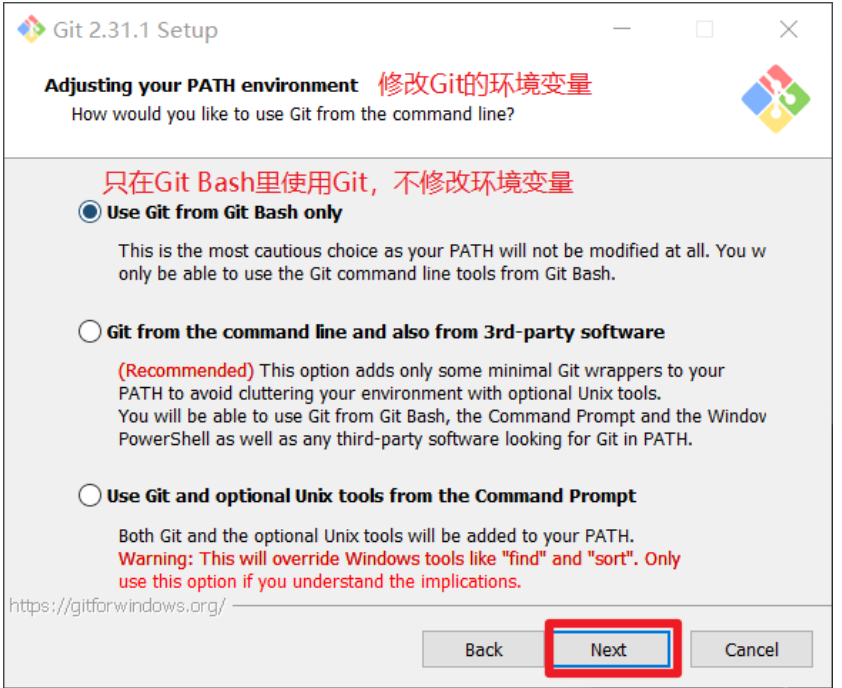
1. Git的默认编辑器，建议使用默认的Vim编辑器，然后点击下一步



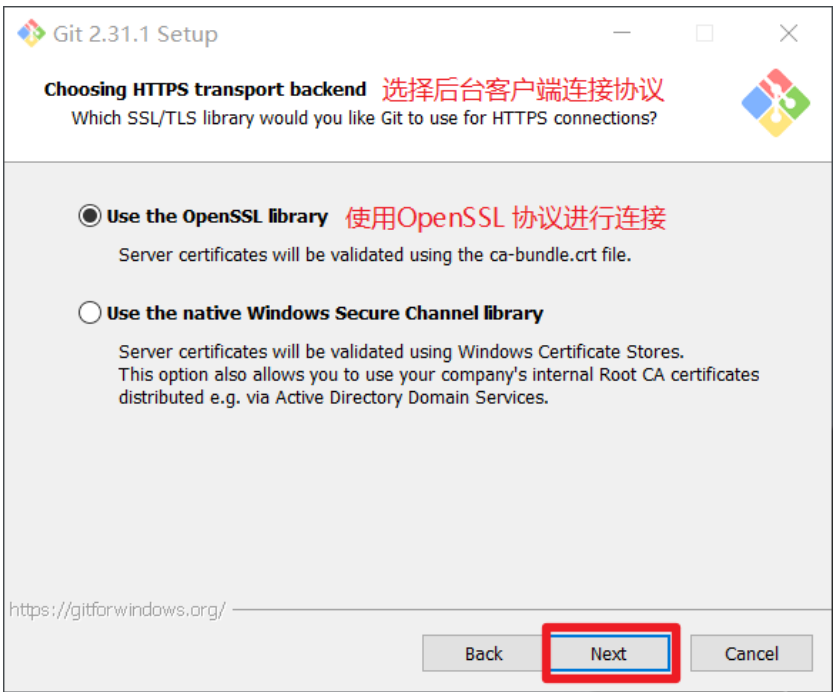
1. 默认分支名的设置，选择让Git决定，分支名默认为master,下一步



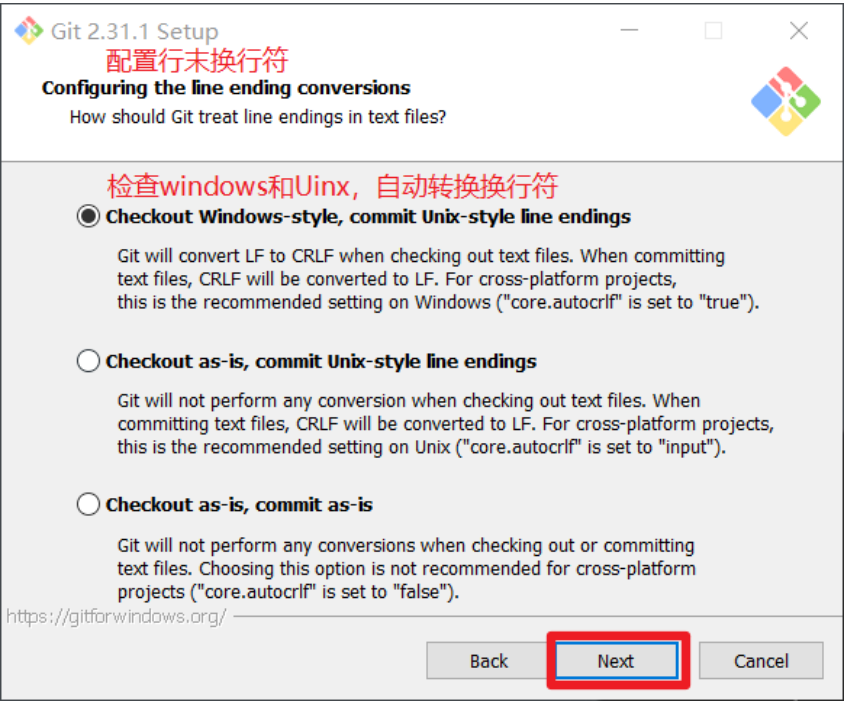
1. 修改Git的环境变量，选择第一个，不修改环境变量，只有在Git Bash中使用



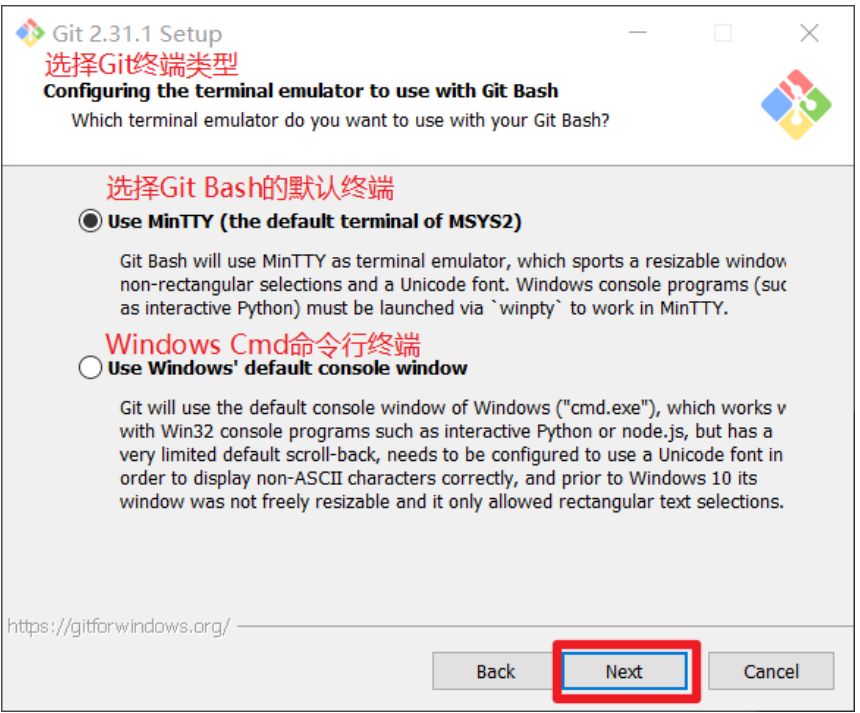
1. 选择后台客户端的连接协议，选择默认的OpenSSL,然后点击下一步



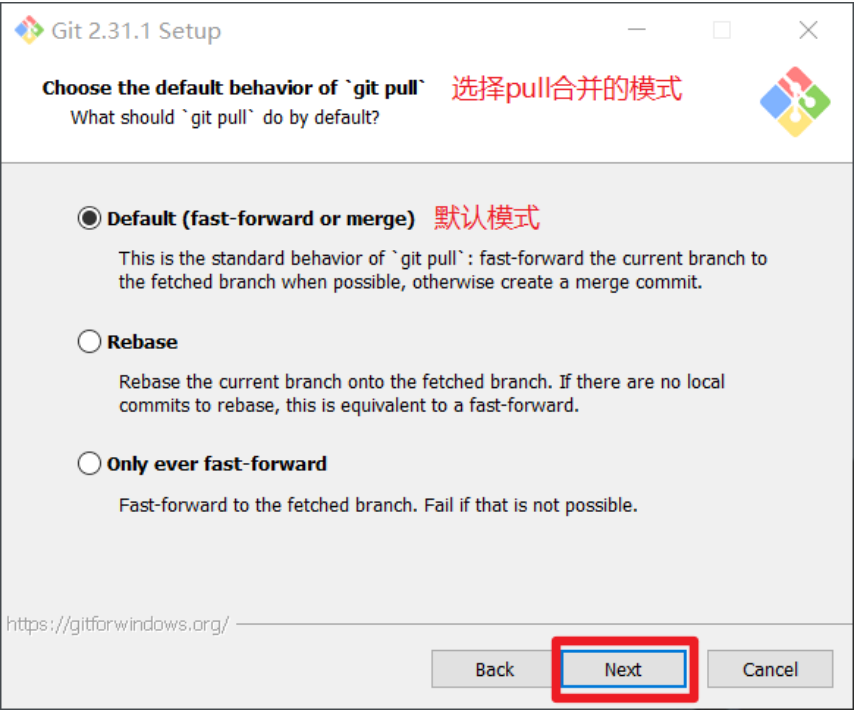
1. 配置Git 文件末行的换行文件符，Windows使用CRLF，Linux使用LF，选择第一个自动转换，然后继续下一步。



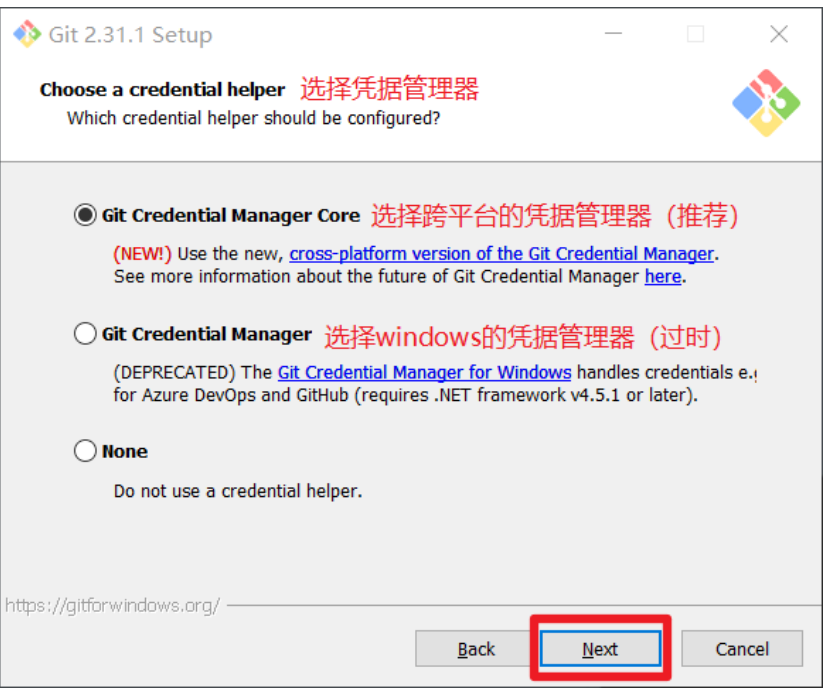
1. 选择Git终端类型，选择默认的Git Bash终端，然后继续下一步



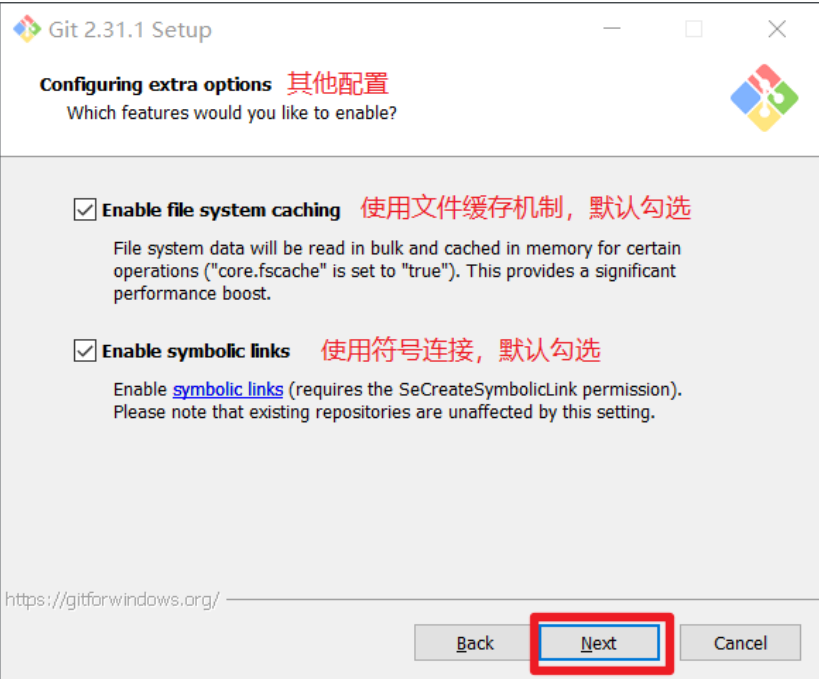
1. 选择Git pull合并的模式，选择默认，然后下一步



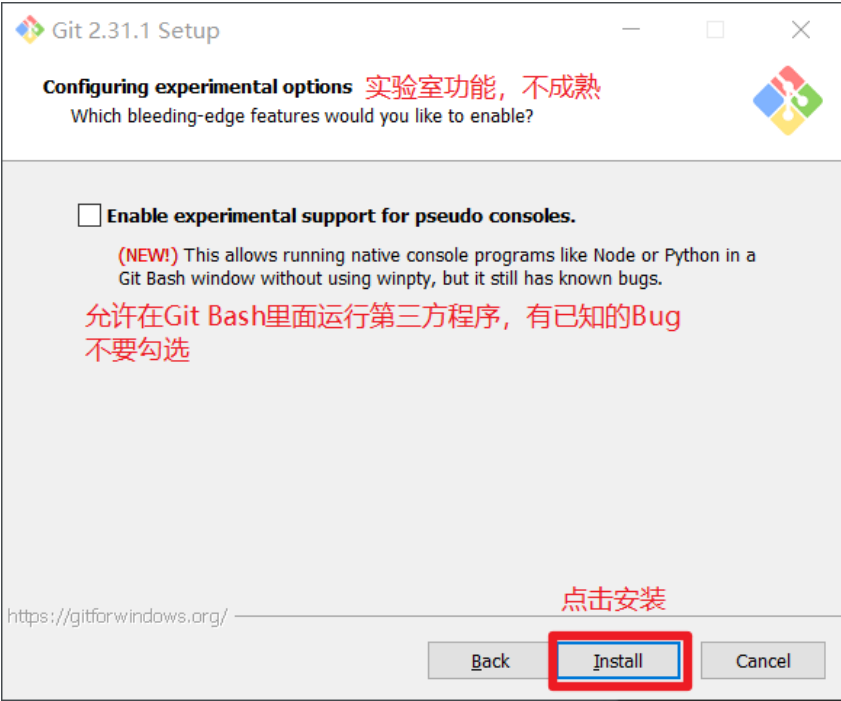
1. 选择Git的凭据管理器，选择默认的跨平台的凭据管理器，然后下一步



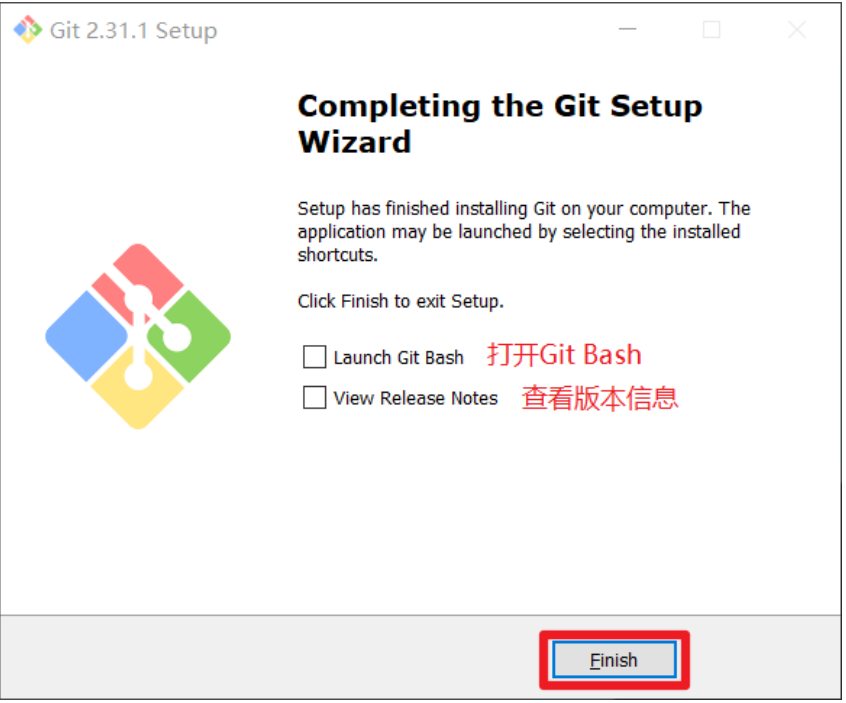
1. 其他配置，选择默认设置，然后下一步



1. 实验室功能，技术还不成熟，有已知的bug，不要勾选，然后点击右下角的Install按钮，开始安装Git。



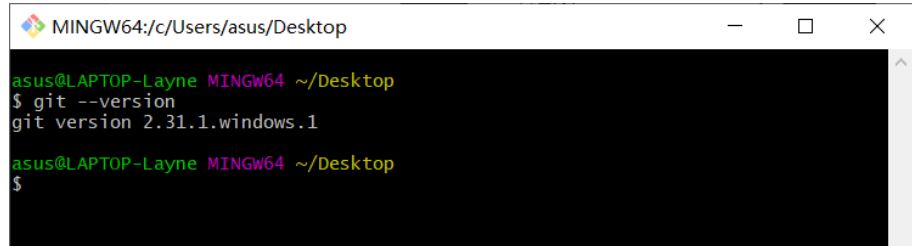
1. 点击Finsh按钮，Git安装成功！



1. 右键任意位置，在右键菜单里选择Git Bash Here即可打开Git Bash命令行终端



1. 在Git Bash终端里输入git --version查看git版本，如图所示，说明Git安装成功



# Git常用命令

## Git的常用命令

|  |  |
| --- | --- |
| **命令名称** | **作用** |
| git config --global user.name 用户名 | 设置用户签名 |
| git config --global user.email 邮箱 | 设置用户邮箱 |
| git init | 初始化本地库 |
| git status | 查看本地库状态 |
| git add 文件名 | 添加到暂存区 |
| git commit -m "日志信息" 文件名 | 提交到本地库 |
| git reflog | 查看历史记录 |
| git reset --hard 版本号 | 版本穿梭 |

## 设置用户名

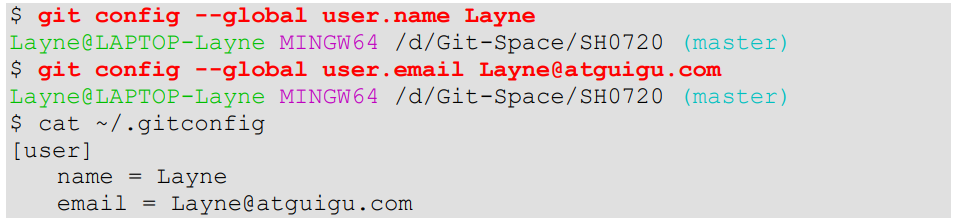
### 2.1基本语法

git config --global user.name 用户名

git config --global user.email 邮箱

### 2.2案例实操

全局范围的签名设置：图例设置的用户名为：Layne 全局邮箱为：[Layne@atguigu.com](mailto:Layne@atguigu.com)

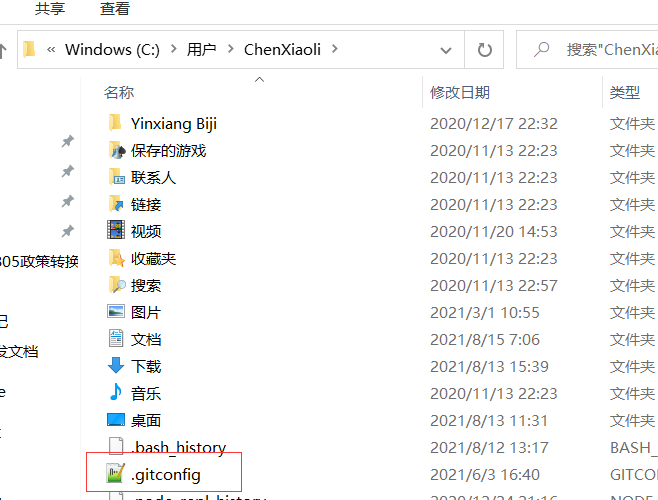


说明：

签名的作用是区分不同操作者身份。用户的签名信息在每一个版本的提交信息中能够看到，以此确认本次提交是谁做的。Git首次安装必须设置一下用户签名，否则无法提交代码。

※注意：这里设置用户签名和将来登录GitHub（或其他代码托管中心）的账号没有任何关系。

### 2.3结果查看



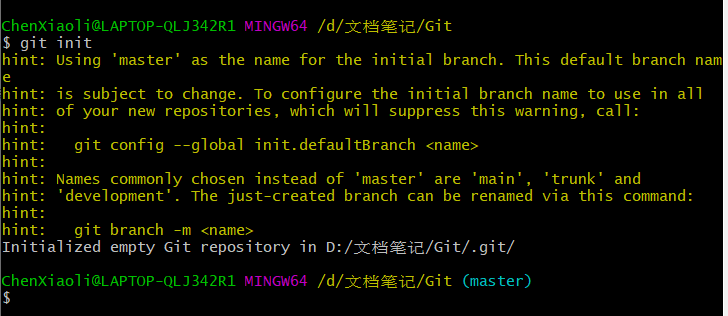
## 初始化本地库

### 3.1基本语法

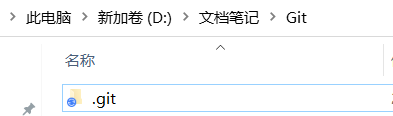
git init

### 3.2实战案例

在进入需要处理工作空间中，右击选择Git Bash Here



### 3.3结果查看



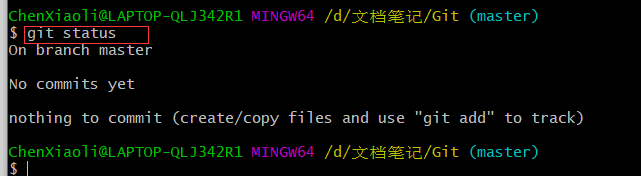
## 查看本地库状态

### 4.1基本语法

git status

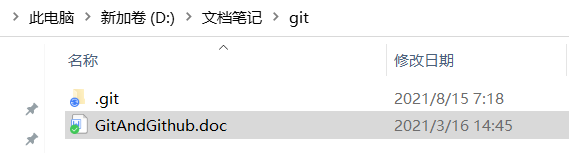
### 4.2实现案例

#### 4.2.1首次查看（工作区没有任何文件）

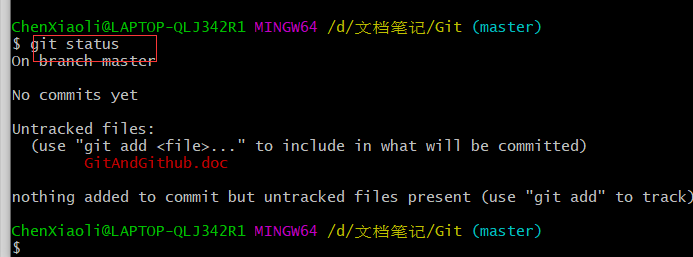


#### 4.2.2新增文件

将文件GitAndGithub.doc存放在路径D:\文档笔记\Git



#### 4.2.3再次查看



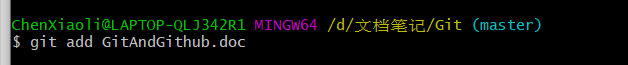
## 添加暂存区

### 5.1将工作区文件添加到暂存区

#### 5.1.1基本语法

git add 文件名

#### 5.1.2案例操作



### 5.2查看状态（检测到暂存区有新文件）



## 提交本地库

### 6.1将暂存区的文件提交大本地库

#### 6.1.1基本语法

git commit -m "日志信息" 文件名

#### 6.1.2案例实操

ChenXiaoli@LAPTOP-QLJ342R1 MINGW64 /d/文档笔记/Git (master)

$ git commit -m "创建第一版文档" GitAndGithub.doc

[master (root-commit) 1b53da0] 创建第一版文档

1 file changed, 0 insertions(+), 0 deletions(-)

create mode 100644 GitAndGithub.doc

### 6.2查看状态（没有文件需要提交）

ChenXiaoli@LAPTOP-QLJ342R1 MINGW64 /d/文档笔记/Git (master)

$ git status

On branch master

nothing to commit, working tree clean

## 修改文件

### 7.1查看状态（检测到工作区有文件被修改）

ChenXiaoli@LAPTOP-QLJ342R1 MINGW64 /d/文档笔记/Git (master)

$ git status

On branch master

Changes not staged for commit:

(use "git add <file>..." to update what will be committed)

(use "git restore <file>..." to discard changes in working directory)

modified: GitAndGithub.doc

no changes added to commit (use "git add" and/or "git commit -a")

### 7.2将修改的文件再次提交到缓存区

ChenXiaoli@LAPTOP-QLJ342R1 MINGW64 /d/文档笔记/Git (master)

$ git add GitAndGithub.doc

### 7.3查看状态（工作区的修改添加到缓存区了）

ChenXiaoli@LAPTOP-QLJ342R1 MINGW64 /d/文档笔记/Git (master)

$ git status

On branch master

Changes to be committed:

(use "git restore --staged <file>..." to unstage)

modified: GitAndGithub.doc

## 历史版本

### 8.1查看历史版本

#### 8.1.1基本语法

git reflog 查看版本信息

git log 查看版本详细信息

#### 8.1.2案例实操

ChenXiaoli@LAPTOP-QLJ342R1 MINGW64 /d/文档笔记/Git (master)

$ git reflog

85379ce (HEAD -> master) HEAD@{0}: commit: 文档提交的第三个版本

2a7178a HEAD@{1}: commit: 提交文件第二版本

1b53da0 HEAD@{2}: commit (initial): 创建第一版文档

ChenXiaoli@LAPTOP-QLJ342R1 MINGW64 /d/文档笔记/Git (master)

$ git log

commit 85379cebcab44a2a491cac8268343407e5739c55 (HEAD -> master)

Author: chenxl21 <chenxl21@\*\*\*\*\*.com>

Date: Sun Aug 15 08:36:15 2021 +0800

文档提交的第三个版本

commit 2a7178afbac1fd121e21a82368db4dfdf5123aae

Author: chenxl21 <chenxl21@\*\*\*\*\*.com>

Date: Sun Aug 15 08:31:04 2021 +0800

提交文件第二版本

commit 1b53da0dc135e0cebabf963346e486dfd0118ae1

Author: chenxl21 <chenxl21@\*\*\*\*\*.com>

Date: Sun Aug 15 07:29:28 2021 +0800

创建第一版文档

### 8.2版本穿梭

#### 8.2.1基本语法

git reset --hard 版本号

#### 8.2.2案例实操

ChenXiaoli@LAPTOP-QLJ342R1 MINGW64 /d/文档笔记/Git (master)

$ git reflog //查看历史版本

85379ce (HEAD -> master) HEAD@{0}: commit: 文档提交的第三个版本

2a7178a HEAD@{1}: commit: 提交文件第二版本

1b53da0 HEAD@{2}: commit (initial): 创建第一版文档

ChenXiaoli@LAPTOP-QLJ342R1 MINGW64 /d/文档笔记/Git (master)

$ git reset --hard 1b53da0 //将版本穿梭到第一版

HEAD is now at 1b53da0 创建第一版文档

ChenXiaoli@LAPTOP-QLJ342R1 MINGW64 /d/文档笔记/Git (master)

$ git reflog //指针发生变换跳转到第一版

1b53da0 (HEAD -> master) HEAD@{0}: reset: moving to 1b53da0

85379ce HEAD@{1}: commit: 文档提交的第三个版本

2a7178a HEAD@{2}: commit: 提交文件第二版本

1b53da0 (HEAD -> master) HEAD@{3}: commit (initial): 创建第一版文档

ChenXiaoli@LAPTOP-QLJ342R1 MINGW64 /d/文档笔记/Git (master)

# Git分支操作

## 什么是分支

在版本控制的过程中，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分支。使用分支就意味着程序员可以把自己的工作从开发主线分离出来。开发自己分支的时候，不会影响主线分支的运行。可以简单理解分支就是一个副本，一个分支就是一个单独副本。（分支的底层其实也是指针的引用）

## 分支的好处

同时并推向多个功能开发，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。

## 分支的操作

|  |  |
| --- | --- |
| **命令名称** | **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的分支上 |

### 3.1查看分支

#### 3.1.1基本语法

git branch -v

#### 3.1.2实例操作

ChenXiaoli@LAPTOP-QLJ342R1 MINGW64 /d/文档笔记/Git (master)

$ git branch -v

\* master 1b53da0 创建第一版文档 //\* master代表当前的分区

ChenXiaoli@LAPTOP-QLJ342R1 MINGW64 /d/文档笔记/Git (master)

### 3.2创建分支

#### 3.2.1基本语法

git branch 分支名

#### 3.2.2实例操作

ChenXiaoli@LAPTOP-QLJ342R1 MINGW64 /d/文档笔记/Git (master)

$ git branch hot-fix //创建名称为 hot-fix

ChenXiaoli@LAPTOP-QLJ342R1 MINGW64 /d/文档笔记/Git (master)

$ git branch -v //查看分支的情况

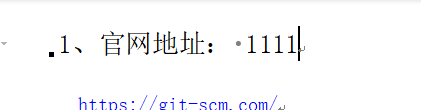
hot-fix 1b53da0 创建第一版文档 //新创建的分支

\* master 1b53da0 创建第一版文档

ChenXiaoli@LAPTOP-QLJ342R1 MINGW64 /d/文档笔记/Git (master)

### 3.3修改分支

1. 修改文档，在标题后添加111



ChenXiaoli@LAPTOP-QLJ342R1 MINGW64 /d/文档笔记/Git (master)

$ git add GitAndGithub.doc //将修改后的文件提交的缓存

ChenXiaoli@LAPTOP-QLJ342R1 MINGW64 /d/文档笔记/Git (master)

$ git commit -m "文档的标题添加三个1" GitAndGithub.doc //将代码提交本地库

[master 7925e2a] 文档的标题添加三个1

1 file changed, 0 insertions(+), 0 deletions(-)

ChenXiaoli@LAPTOP-QLJ342R1 MINGW64 /d/文档笔记/Git (master)

$ git reflog //查看当前的文档的状态

7925e2a (HEAD -> master) HEAD@{0}: commit: 文档的标题添加三个1 //当前的主分支上这//个版本

1b53da0 (hot-fix) HEAD@{1}: reset: moving to 1b53da0

85379ce HEAD@{2}: commit: 文档提交的第三个版本

2a7178a HEAD@{3}: commit: 提交文件第二版本

1b53da0 (hot-fix) HEAD@{4}: commit (initial): 创建第一版文档

ChenXiaoli@LAPTOP-QLJ342R1 MINGW64 /d/文档笔记/Git (master)

### 3.4切换分支

#### 3.4.1基本语法

git checkout 分支名称

#### 3.4.2案例实操

$ git checkout hot-fix //将分支切换到 hot-fix

Switched to branch 'hot-fix'

ChenXiaoli@LAPTOP-QLJ342R1 MINGW64 /d/文档笔记/Git (hot-fix)

$ git reflog //查看分支的当前状态

1b53da0 (HEAD -> hot-fix) HEAD@{0}: checkout: moving from master to hot-fix

7925e2a (master) HEAD@{1}: commit: 文档的标题添加三个1

1b53da0 (HEAD -> hot-fix) HEAD@{2}: reset: moving to 1b53da0

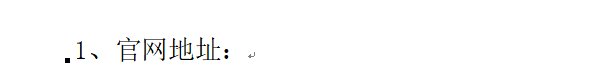
85379ce HEAD@{3}: commit: 文档提交的第三个版本

2a7178a HEAD@{4}: commit: 提交文件第二版本

//当前的版本指向了 hot-fix的分支

1b53da0 (HEAD -> hot-fix) HEAD@{5}: commit (initial): 创建第一版文档

1、查看文档，之前子master分支的修改的内容也不存在了（后面添加的几个1）：如下图



2、在当前的hot-fix版本在第二条添加四个2如图：



3、将在hot-fix修改的内容添加到缓存，并进行提交：

ChenXiaoli@LAPTOP-QLJ342R1 MINGW64 /d/文档笔记/Git (hot-fix)

$ git add GitAndGithub.doc

ChenXiaoli@LAPTOP-QLJ342R1 MINGW64 /d/文档笔记/Git (hot-fix)

$ git commit -m "在hot-fix上修改的第一版添加四个2" GitAndGithub.doc

[hot-fix 7b7c78a] 在hot-fix上修改的第一版添加四个2

1 file changed, 0 insertions(+), 0 deletions(-)

### 3.5合并分支

#### 3.5.1环境的准备

1. 初始化测试

ChenXiaoli@LAPTOP-QLJ342R1 MINGW64 /d/文档笔记/Git测试

$ git init

hint: Using 'master' as the name for the initial branch. This default branch name

hint: is subject to change. To configure the initial branch name to use in all

hint: of your new repositories, which will suppress this warning, call:

hint:

hint: git config --global init.defaultBranch <name>

hint:

hint: Names commonly chosen instead of 'master' are 'main', 'trunk' and

hint: 'development'. The just-created branch can be renamed via this command:

hint:

hint: git branch -m <name>

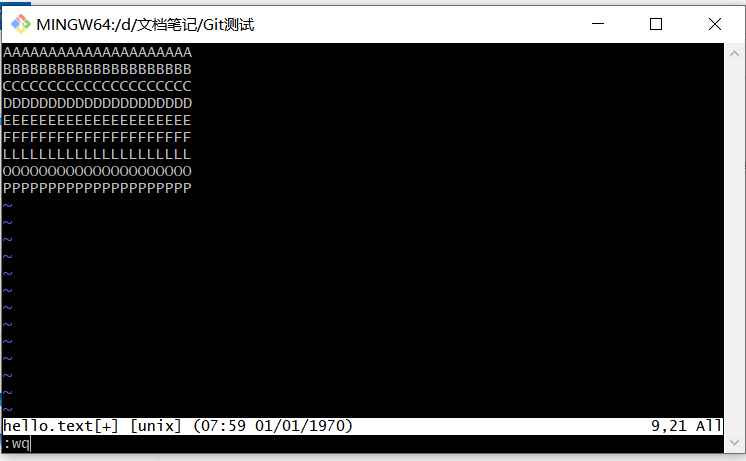
Initialized empty Git repository in D:/文档笔记/Git测试/.git/

ChenXiaoli@LAPTOP-QLJ342R1 MINGW64 /d/文档笔记/Git测试 (master)

2、在初始化的git 仓库下面新建一个 hello.text 文件，创建后直接回车键，进入编辑模式，添加内容后按ESC键，然后录入：wq保存退出

ChenXiaoli@LAPTOP-QLJ342R1 MINGW64 /d/文档笔记/Git测试 (master)

$ vim hello.text



3、查看状态 git status

ChenXiaoli@LAPTOP-QLJ342R1 MINGW64 /d/文档笔记/Git测试 (master)

$ git status

On branch master

No commits yet

Untracked files:

(use "git add <file>..." to include in what will be committed)

hello.text

nothing added to commit but untracked files present (use "git add" to track)

1. 添加缓存：git add hello.text

ChenXiaoli@LAPTOP-QLJ342R1 MINGW64 /d/文档笔记/Git测试 (master)

$ git add hello.text

warning: LF will be replaced by CRLF in hello.text.

The file will have its original line endings in your working directory

ChenXiaoli@LAPTOP-QLJ342R1 MINGW64 /d/文档笔记/Git测试 (master)

1. 提交 git commit -m "master first commit" hello.text

ChenXiaoli@LAPTOP-QLJ342R1 MINGW64 /d/文档笔记/Git测试 (master)

$ git commit -m "master first commit" hello.text

warning: LF will be replaced by CRLF in hello.text.

The file will have its original line endings in your working directory

[master (root-commit) 0b69504] master first commit

1 file changed, 9 insertions(+)

create mode 100644 hello.text

进入文件 点击i 在第一行添加几个1，保存退出，并且按照上面步添加缓存，提交

1. 创建分支并切换分支

ChenXiaoli@LAPTOP-QLJ342R1 MINGW64 /d/文档笔记/Git测试 (master)

//（1）查看当前的分支

$ git branch -v

\* master 55fbe3c second master commit

ChenXiaoli@LAPTOP-QLJ342R1 MINGW64 /d/文档笔记/Git测试 (master)

//（2）创建分支 分支名为 hot-fix

$ git branch hot-fix

ChenXiaoli@LAPTOP-QLJ342R1 MINGW64 /d/文档笔记/Git测试 (master)

//（3）查看当前的分支，新创建的分支已经存在

$ git branch -v

hot-fix 55fbe3c second master commit

\* master 55fbe3c second master commit

ChenXiaoli@LAPTOP-QLJ342R1 MINGW64 /d/文档笔记/Git测试 (master)

//（4）切换到当前的分支 hot-fix

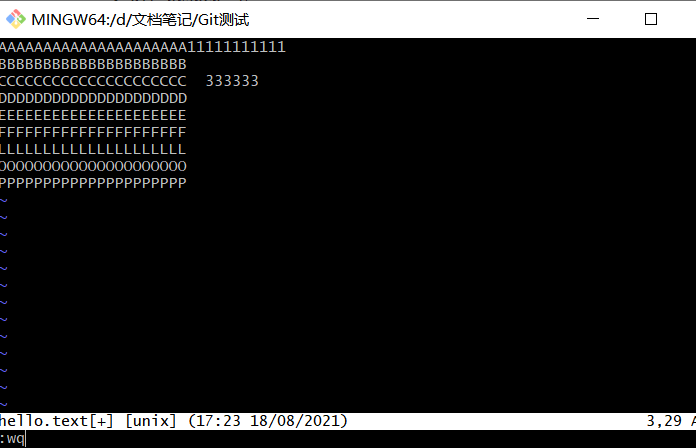
$ git checkout hot-fix

Switched to branch 'hot-fix'

ChenXiaoli@LAPTOP-QLJ342R1 MINGW64 /d/文档笔记/Git测试 (hot-fix)

$

1. 修改分支



在分支第三行添加几个3 ，在分支下添加缓存并提交

ChenXiaoli@LAPTOP-QLJ342R1 MINGW64 /d/文档笔记/Git测试 (hot-fix)

//（1）查案当前文档的状态

$ git status

On branch hot-fix

Changes not staged for commit:

(use "git add <file>..." to update what will be committed)

(use "git restore <file>..." to discard changes in working directory)

modified: hello.text

no changes added to commit (use "git add" and/or "git commit -a")

ChenXiaoli@LAPTOP-QLJ342R1 MINGW64 /d/文档笔记/Git测试 (hot-fix)

//（2）将当前的文档提交到缓存区

$ git add hello.text

warning: LF will be replaced by CRLF in hello.text.

The file will have its original line endings in your working directory

ChenXiaoli@LAPTOP-QLJ342R1 MINGW64 /d/文档笔记/Git测试 (hot-fix)

//（3）提交当前的文档

$ git commit -m "first hot-fix commit" hello.text

warning: LF will be replaced by CRLF in hello.text.

The file will have its original line endings in your working directory

[hot-fix ae5c31f] first hot-fix commit

1 file changed, 1 insertion(+), 1 deletion(-)

ChenXiaoli@LAPTOP-QLJ342R1 MINGW64 /d/文档笔记/Git测试 (hot-fix)

//（4）查看当期文档的状态

$ git status

On branch hot-fix

nothing to commit, working tree clean

ChenXiaoli@LAPTOP-QLJ342R1 MINGW64 /d/文档笔记/Git测试 (hot-fix)

//（5）查看当前分支文档的内容

$ cat hello.text

AAAAAAAAAAAAAAAAAAAAA11111111111

BBBBBBBBBBBBBBBBBBBBB

CCCCCCCCCCCCCCCCCCCCC 333333

DDDDDDDDDDDDDDDDDDDDD

EEEEEEEEEEEEEEEEEEEEE

FFFFFFFFFFFFFFFFFFFFF

LLLLLLLLLLLLLLLLLLLLL

OOOOOOOOOOOOOOOOOOOOO

PPPPPPPPPPPPPPPPPPPPP

ChenXiaoli@LAPTOP-QLJ342R1 MINGW64 /d/文档笔记/Git测试 (hot-fix)

//（6）切换的master分支

$ git checkout master

Switched to branch 'master'

ChenXiaoli@LAPTOP-QLJ342R1 MINGW64 /d/文档笔记/Git测试 (master)

//（7）查看master分支下文档，在hot-fix上修改内容没有展示，所以后面需要分支的合并

$ cat hello.text

AAAAAAAAAAAAAAAAAAAAA11111111111

BBBBBBBBBBBBBBBBBBBBB

CCCCCCCCCCCCCCCCCCCCC

DDDDDDDDDDDDDDDDDDDDD

EEEEEEEEEEEEEEEEEEEEE

FFFFFFFFFFFFFFFFFFFFF

LLLLLLLLLLLLLLLLLLLLL

OOOOOOOOOOOOOOOOOOOOO

PPPPPPPPPPPPPPPPPPPPP

ChenXiaoli@LAPTOP-QLJ342R1 MINGW64 /d/文档笔记/Git测试 (master)

#### 3.5.2基本语法

git merge 分支名称

#### 3.5.3案例实操

ChenXiaoli@LAPTOP-QLJ342R1 MINGW64 /d/文档笔记/Git测试 (master)

//（1）将hot-fix分支切换到master分支，需要切换到master分支

$ git checkout master

Already on 'master'

ChenXiaoli@LAPTOP-QLJ342R1 MINGW64 /d/文档笔记/Git测试 (master)

//（2）执行分支合并命令，其中分支名为需要合并的hot-fix

$ git merge hot-fix

Updating 55fbe3c..ae5c31f

Fast-forward

hello.text | 2 +-

1 file changed, 1 insertion(+), 1 deletion(-)

ChenXiaoli@LAPTOP-QLJ342R1 MINGW64 /d/文档笔记/Git测试 (master)

//（3）查看文档在hot-fix上面新增的内容已经合并到master分支了

$ cat hello.text

AAAAAAAAAAAAAAAAAAAAA11111111111

BBBBBBBBBBBBBBBBBBBBB

CCCCCCCCCCCCCCCCCCCCC 333333

DDDDDDDDDDDDDDDDDDDDD

EEEEEEEEEEEEEEEEEEEEE

FFFFFFFFFFFFFFFFFFFFF

LLLLLLLLLLLLLLLLLLLLL

OOOOOOOOOOOOOOOOOOOOO

PPPPPPPPPPPPPPPPPPPPP

ChenXiaoli@LAPTOP-QLJ342R1 MINGW64 /d/文档笔记/Git测试 (master)

### 3.6产生冲突

1. 在master分支打开文件，在最后一行编辑添加数字6，添加缓存并提交

ChenXiaoli@LAPTOP-QLJ342R1 MINGW64 /d/文档笔记/Git测试 (master)

$ cat hello.text

AAAAAAAAAAAAAAAAAAAAA11111111111

BBBBBBBBBBBBBBBBBBBBB

CCCCCCCCCCCCCCCCCCCCC 333333

DDDDDDDDDDDDDDDDDDDDD

EEEEEEEEEEEEEEEEEEEEE

FFFFFFFFFFFFFFFFFFFFF

LLLLLLLLLLLLLLLLLLLLL

OOOOOOOOOOOOOOOOOOOOO

PPPPPPPPPPPPPPPPPPPPP 666666666666

1. 在hot-fix分支上打开文件修改倒数第二行添加数字8，添加到缓存并提交

ChenXiaoli@LAPTOP-QLJ342R1 MINGW64 /d/文档笔记/Git测试 (hot-fix)

$ cat hello.text

AAAAAAAAAAAAAAAAAAAAA11111111111

BBBBBBBBBBBBBBBBBBBBB

CCCCCCCCCCCCCCCCCCCCC 333333

DDDDDDDDDDDDDDDDDDDDD

EEEEEEEEEEEEEEEEEEEEE

FFFFFFFFFFFFFFFFFFFFF

LLLLLLLLLLLLLLLLLLLLL

OOOOOOOOOOOOOOOOOOOOO 8888

PPPPPPPPPPPPPPPPPPPPP

ChenXiaoli@LAPTOP-QLJ342R1 MINGW64 /d/文档笔记/Git测试 (hot-fix)

1. 切换master分支合并hot-fix分支的内容冲突出现

ChenXiaoli@LAPTOP-QLJ342R1 MINGW64 /d/文档笔记/Git测试 (hot-fix)

//（1）切换到主分支

$ git checkout master

Switched to branch 'master'

ChenXiaoli@LAPTOP-QLJ342R1 MINGW64 /d/文档笔记/Git测试 (master)

//（2）在主分支上合并hot-fix分支这时冲突出现

$ git merge hot-fix

Auto-merging hello.text

CONFLICT (content): Merge conflict in hello.text

Automatic merge failed; fix conflicts and then commit the result.

ChenXiaoli@LAPTOP-QLJ342R1 MINGW64 /d/文档笔记/Git测试 (master|MERGING)

### 3.7解决冲突

#### 3.7.1 产生的冲突

ChenXiaoli@LAPTOP-QLJ342R1 MINGW64 /d/文档笔记/Git测试 (master|MERGING)

$ cat hello.text

AAAAAAAAAAAAAAAAAAAAA11111111111

BBBBBBBBBBBBBBBBBBBBB

CCCCCCCCCCCCCCCCCCCCC 333333

DDDDDDDDDDDDDDDDDDDDD

EEEEEEEEEEEEEEEEEEEEE

FFFFFFFFFFFFFFFFFFFFF

LLLLLLLLLLLLLLLLLLLLL

<<<<<<< HEAD

OOOOOOOOOOOOOOOOOOOOO

PPPPPPPPPPPPPPPPPPPPP 666666666666

=======

OOOOOOOOOOOOOOOOOOOOO 8888

PPPPPPPPPPPPPPPPPPPPP

>>>>>>> hot-fix

ChenXiaoli@LAPTOP-QLJ342R1 MINGW64 /d/文档笔记/Git测试 (master|MERGING)

#### 3.7.2产生冲突的原因

合并分支的时候，两个分支在同一个文件的同一个位置有两完全不同的修改，git无法替我们决定使用哪一个版本，必须人为决定使用哪一套代码。查看状态发现两个分支都有代码的修改：

ChenXiaoli@LAPTOP-QLJ342R1 MINGW64 /d/文档笔记/Git测试 (master|MERGING)

$ git status

On branch master

You have unmerged paths.

(fix conflicts and run "git commit")

(use "git merge --abort" to abort the merge)

Unmerged paths:

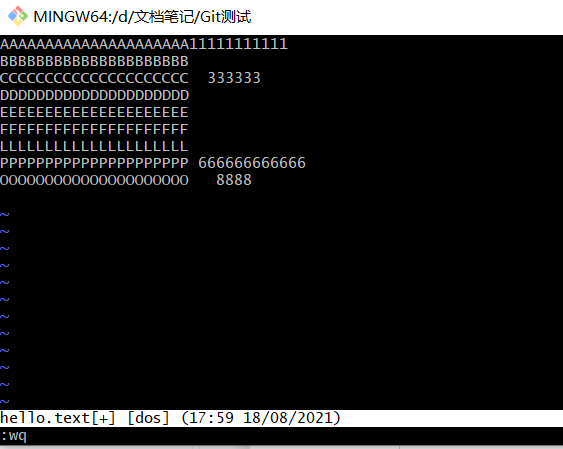
(use "git add <file>..." to mark resolution)

both modified: hello.text

no changes added to commit (use "git add" and/or "git commit -a")

#### 3.7.3解决冲突

1. 编辑有冲突的文件，删除特殊符号，决定要使用的内容 特殊符号：<<<<<<< HEAD 当前分支的代码 ======= 合并过来的代码 >>>>>>> hot-fix



1. 添加到缓存区

ChenXiaoli@LAPTOP-QLJ342R1 MINGW64 /d/文档笔记/Git测试 (master|MERGING)

$ git add hello.text

1. 执行提交（注意：此时使用git commit命令时不能带文件名）

ChenXiaoli@LAPTOP-QLJ342R1 MINGW64 /d/文档笔记/Git测试 (master|MERGING)

$ git commit -m "merge hot-fix"

[master d222796] merge hot-fix

//master 后面的|MERGING 消失了

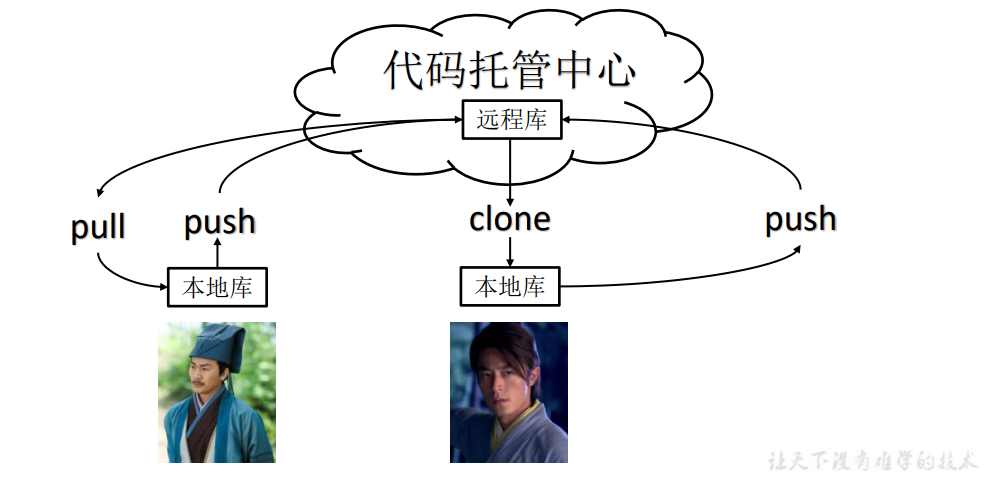
ChenXiaoli@LAPTOP-QLJ342R1 MINGW64 /d/文档笔记/Git测试 (master)

## 创建分支切换分支

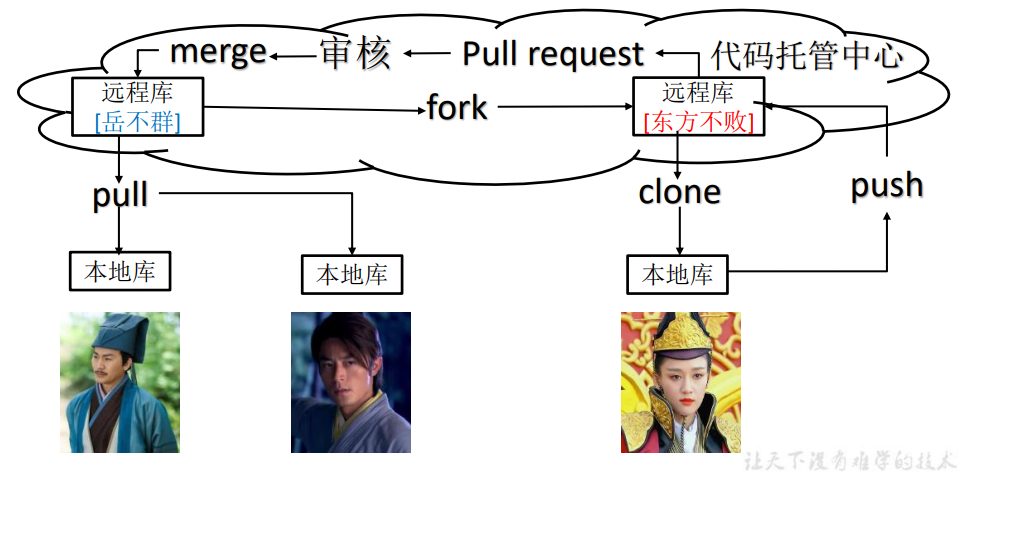
master、hot-fix其实都是指向具体版本记录的指针。当前所在的分支，其实是由HEAD决定的。所以创建分支的本质就是多创建一个指针。 HEAD如果指向master，那么我们现在就在master分支上。 HEAD如果执行hotfix，那么我们现在就在hotfix分支上。

# Git团队协作机制

1. 团队内协作



1. 跨团队外协作

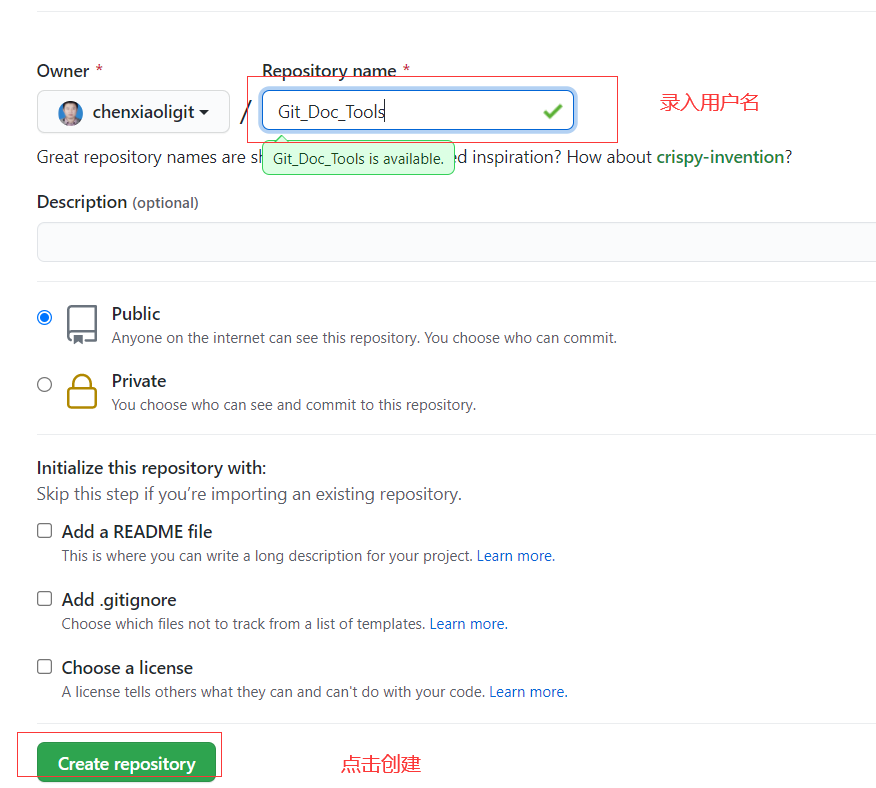
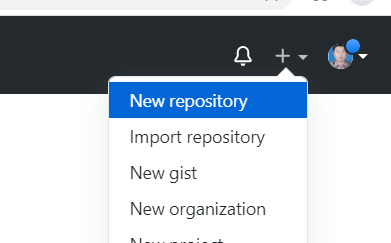


# GitHub操作

## 网址账号信息

1. 网址信息：<https://github.com/>
2. 用户名：[531088243@qq.com](mailto:531088243@qq.com)
3. 密码：Cxl15251892818

## 创建远程库



## 远程库操作

### 3.1常用的操作命令

|  |  |
| --- | --- |
| 命令 | 作用： |
| git remote -v | 查看当前所有远程库别名 |
| git remote add 别名 远程库地址 | 起别名 |
| git push 别名 分支 | 推送本地分支上内容到远程库 |
| git clone 远程地址 | 将远程库的内容克隆到本地 |
| git pull 远程库地址别名 远程分支名 | 将远程仓库对于分支最新的内容拉下来后与当前的分支直接合并 |

### 3.2创建远程库别名

#### 3.2.1基本语法

1. git remote -v 查看当前所有远程地址别名
2. git remote add 别名 远程地址

#### 3.2.2案例实操

ChenXiaoli@LAPTOP-QLJ342R1 MINGW64 /d/文档笔记/Git测试 (master)

//查看当前所欲远程库别名，目前没创建所以还不存在

$ git remote -v

ChenXiaoli@LAPTOP-QLJ342R1 MINGW64 /d/文档笔记/Git测试 (master)

ChenXiaoli@LAPTOP-QLJ342R1 MINGW64 /d/文档笔记/Git\_Doc\_Tools (master)

//创建别名为 ori的远程库 后面的地址为远程库地址

$ git remote add ori https://github.com/chenxiaoligit/Git\_Doc\_Tools.git

ChenXiaoli@LAPTOP-QLJ342R1 MINGW64 /d/文档笔记/Git\_Doc\_Tools (master)

ChenXiaoli@LAPTOP-QLJ342R1 MINGW64 /d/文档笔记/Git\_Doc\_Tools (master)

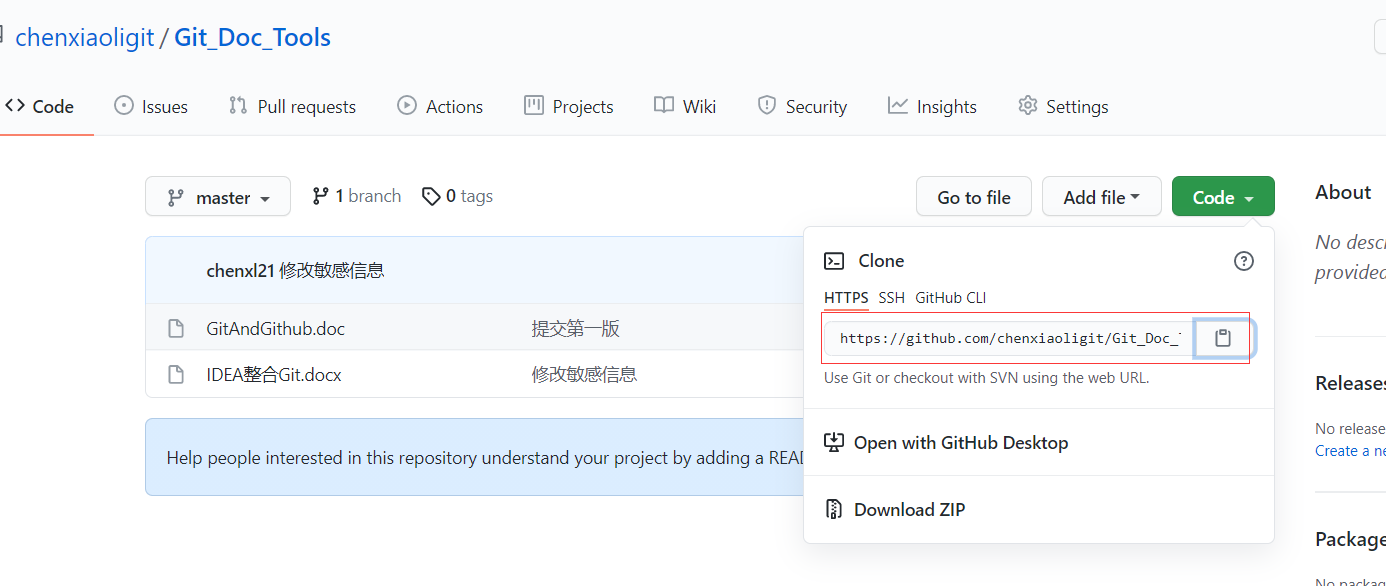
//再次查看远程库已经存在

$ git remote -v

ori https://github.com/chenxiaoligit/Git\_Doc\_Tools.git (fetch)

ori https://github.com/chenxiaoligit/Git\_Doc\_Tools.git (push)

<https://github.com/chenxiaoligit/Git_Doc_Tools.git> 的地址就是生成远程库对应的地址，具体的如下图所示：



### 3.3推送本地分支到远程库

ChenXiaoli@LAPTOP-QLJ342R1 MINGW64 /d/文档笔记/Git\_Doc\_Tools (master)

$ git remote add ori https://github.com/chenxiaoligit/Git\_Doc\_Tools.git

ChenXiaoli@LAPTOP-QLJ342R1 MINGW64 /d/文档笔记/Git\_Doc\_Tools (master)

$ git remote -v

ori https://github.com/chenxiaoligit/Git\_Doc\_Tools.git (fetch)

ori https://github.com/chenxiaoligit/Git\_Doc\_Tools.git (push)

ChenXiaoli@LAPTOP-QLJ342R1 MINGW64 /d/文档笔记/Git\_Doc\_Tools (master)

$ gitremote -v

bash: gitremote: command not found

ChenXiaoli@LAPTOP-QLJ342R1 MINGW64 /d/文档笔记/Git\_Doc\_Tools (master)

$ git removte add ori https://github.com/chenxiaoligit/Git\_Doc\_Tools.git

git: 'removte' is not a git command. See 'git --help'.

The most similar command is

remote

ChenXiaoli@LAPTOP-QLJ342R1 MINGW64 /d/文档笔记/Git\_Doc\_Tools (master)

$ gitremote -v

bash: gitremote: command not found

ChenXiaoli@LAPTOP-QLJ342R1 MINGW64 /d/文档笔记/Git\_Doc\_Tools (master)

$ git remote add ori https://github.com/chenxiaoligit/Git\_Doc\_Tools.git

ChenXiaoli@LAPTOP-QLJ342R1 MINGW64 /d/文档笔记/Git\_Doc\_Tools (master)

$ gitremote -v

bash: gitremote: command not found

ChenXiaoli@LAPTOP-QLJ342R1 MINGW64 /d/文档笔记/Git\_Doc\_Tools (master)

$ git remote -v

ori https://github.com/chenxiaoligit/Git\_Doc\_Tools.git (fetch)

ori https://github.com/chenxiaoligit/Git\_Doc\_Tools.git (push)

ChenXiaoli@LAPTOP-QLJ342R1 MINGW64 /d/文档笔记/Git\_Doc\_Tools (master)

$

ChenXiaoli@LAPTOP-QLJ342R1 MINGW64 /d/文档笔记/Git\_Doc\_Tools (master)

$

ChenXiaoli@LAPTOP-QLJ342R1 MINGW64 /d/文档笔记/Git\_Doc\_Tools (master)

$

ChenXiaoli@LAPTOP-QLJ342R1 MINGW64 /d/文档笔记/Git\_Doc\_Tools (master)

$

ChenXiaoli@LAPTOP-QLJ342R1 MINGW64 /d/文档笔记/Git\_Doc\_Tools (master)

$

ChenXiaoli@LAPTOP-QLJ342R1 MINGW64 /d/文档笔记/Git\_Doc\_Tools (master)

$ git push ori master

remote: Support for password authentication was removed on August 13, 2021. Please use a personal access token instead.

remote: Please see https://github.blog/2020-12-15-token-authentication-requirements-for-git-operations/ for more information.

fatal: Authentication failed for 'https://github.com/chenxiaoligit/Git\_Doc\_Tools.git/'

ChenXiaoli@LAPTOP-QLJ342R1 MINGW64 /d/文档笔记/Git\_Doc\_Tools (master)

$ git push ori master

fatal: unable to access 'https://github.com/chenxiaoligit/Git\_Doc\_Tools.git/': OpenSSL SSL\_read: Connection was aborted, errno 10053

ChenXiaoli@LAPTOP-QLJ342R1 MINGW64 /d/文档笔记/Git\_Doc\_Tools (master)

$ git push ori master

fatal: unable to access 'https://github.com/chenxiaoligit/Git\_Doc\_Tools.git/': OpenSSL SSL\_read: Connection was aborted, errno 10053

ChenXiaoli@LAPTOP-QLJ342R1 MINGW64 /d/文档笔记/Git\_Doc\_Tools (master)

$ git config --global http.sslVerify false

ChenXiaoli@LAPTOP-QLJ342R1 MINGW64 /d/文档笔记/Git\_Doc\_Tools (master)

$ git push ori master

remote: Invalid username or password.

fatal: Authentication failed for 'https://github.com/chenxiaoligit/Git\_Doc\_Tools.git/'

ChenXiaoli@LAPTOP-QLJ342R1 MINGW64 /d/文档笔记/Git\_Doc\_Tools (master)

$ git push ori master

remote: Invalid username or password.

fatal: Authentication failed for 'https://github.com/chenxiaoligit/Git\_Doc\_Tools.git/'

ChenXiaoli@LAPTOP-QLJ342R1 MINGW64 /d/文档笔记/Git\_Doc\_Tools (master)

$ git push ori master

remote: Invalid username or password.

fatal: Authentication failed for 'https://github.com/chenxiaoligit/Git\_Doc\_Tools.git/'

ChenXiaoli@LAPTOP-QLJ342R1 MINGW64 /d/文档笔记/Git\_Doc\_Tools (master)

$ git push ori master

remote: Invalid username or password.

fatal: Authentication failed for 'https://github.com/chenxiaoligit/Git\_Doc\_Tools.git/'

ChenXiaoli@LAPTOP-QLJ342R1 MINGW64 /d/文档笔记/Git\_Doc\_Tools (master)

$ git push ori master

remote: Support for password authentication was removed on August 13, 2021. Please use a personal access token instead.

remote: Please see https://github.blog/2020-12-15-token-authentication-requirements-for-git-operations/ for more information.

fatal: Authentication failed for 'https://github.com/chenxiaoligit/Git\_Doc\_Tools.git/'

ChenXiaoli@LAPTOP-QLJ342R1 MINGW64 /d/文档笔记/Git\_Doc\_Tools (master)

$ git push ori master

Enumerating objects: 6, done.

Counting objects: 100% (6/6), done.

Delta compression using up to 12 threads

Compressing objects: 100% (5/5), done.

Writing objects: 100% (6/6), 12.85 MiB | 24.87 MiB/s, done.

Total 6 (delta 0), reused 0 (delta 0), pack-reused 0

To https://github.com/chenxiaoligit/Git\_Doc\_Tools.git

\* [new branch] master -> master

ChenXiaoli@LAPTOP-QLJ342R1 MINGW64 /d/文档笔记/Git\_Doc\_Tools (master)

### 3.4克隆远程仓库到本地

### 3.5邀请假如团队

### 3.6拉取远程库内容

## 跨团队协作

## SSH免密登录

ChenXiaoli@LAPTOP-QLJ342R1 MINGW64 /d/文档笔记/Git相关文档和工具 (master)

//查看当前的别名

$ git remote -v

ChenXiaoli@LAPTOP-QLJ342R1 MINGW64 /d/文档笔记/Git相关文档和工具 (master)

//创建别名

$ git remote add Git\_Doc\_Tools https://github.com/chenxiaoligit/Git\_Doc\_Tools.git

ChenXiaoli@LAPTOP-QLJ342R1 MINGW64 /d/文档笔记/Git相关文档和工具 (master)

//查看已创建的别名已经存在

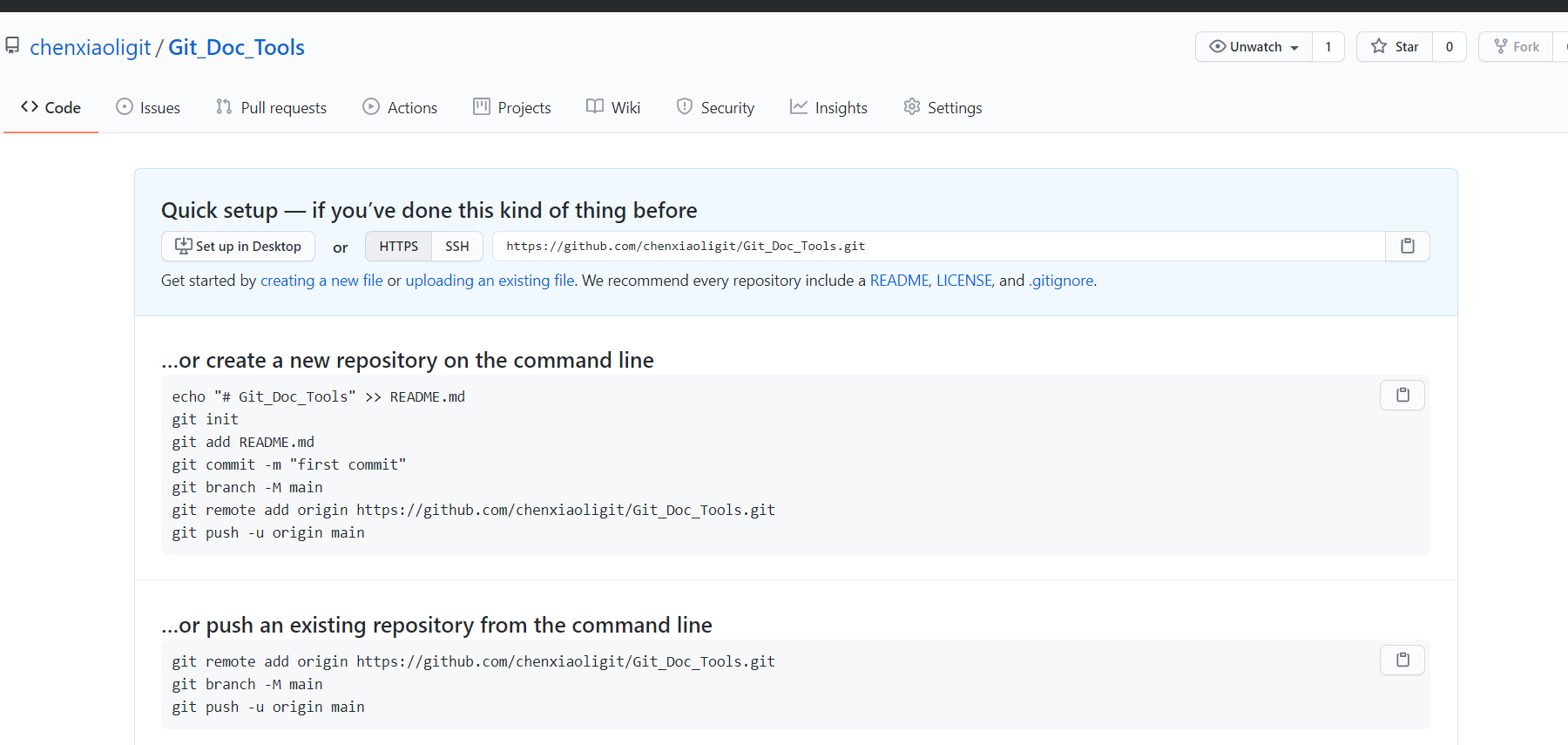
$ git remote -v

Git\_Doc\_Tools https://github.com/chenxiaoligit/Git\_Doc\_Tools.git (fetch)

Git\_Doc\_Tools https://github.com/chenxiaoligit/Git\_Doc\_Tools.git (push)

ChenXiaoli@LAPTOP-QLJ342R1 MINGW64 /d/文档笔记/Git相关文档和工具 (master)

$



ChenXiaoli@LAPTOP-QLJ342R1 MINGW64 /d/文档笔记/Git相关文档和工具 (master)

$ git push Git\_Doc\_Tools master

remote: Support for password authentication was removed on August 13, 2021. Please use a personal access token instead.

remote: Please see https://github.blog/2020-12-15-token-authentication-requirements-for-git-operations/ for more information.

fatal: unable to access 'https://github.com/chenxiaoligit/Git\_Doc\_Tools.git/': The requested URL returned error: 403

ChenXiaoli@LAPTOP-QLJ342R1 MINGW64 /d/文档笔记/Git相关文档和工具 (master)

ChenXiaoli@LAPTOP-QLJ342R1 MINGW64 /d/文档笔记/Git相关文档和工具 (master)

$ git push Git\_Doc\_Tools master

fatal: unable to access 'https://github.com/chenxiaoligit/Git\_Doc\_Tools.git/': OpenSSL SSL\_read: Connection was aborted, errno 10053

ChenXiaoli@LAPTOP-QLJ342R1 MINGW64 /d/文档笔记/Git相关文档和工具 (master)

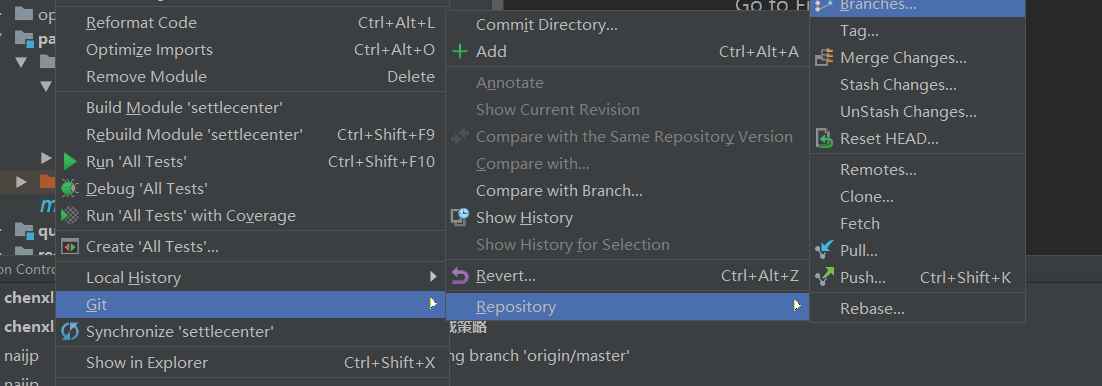
$ git config --global http.postBuffer 524288000

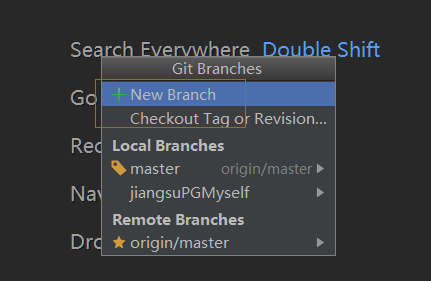
# IDEA集成Git

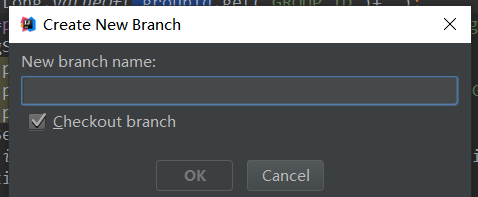
## 7.11创建合并分支案例

### 7.11.1 创建分支

选择Git ==>Repository==>Branches==>New Branch 录入分支的名称点击创建即可

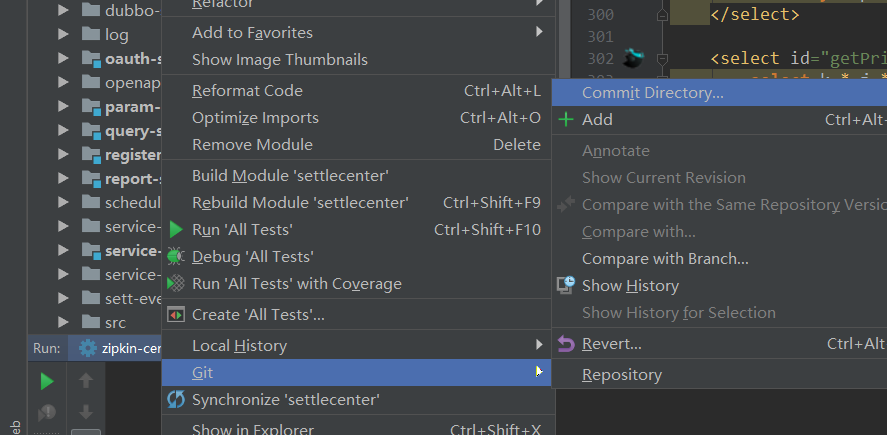


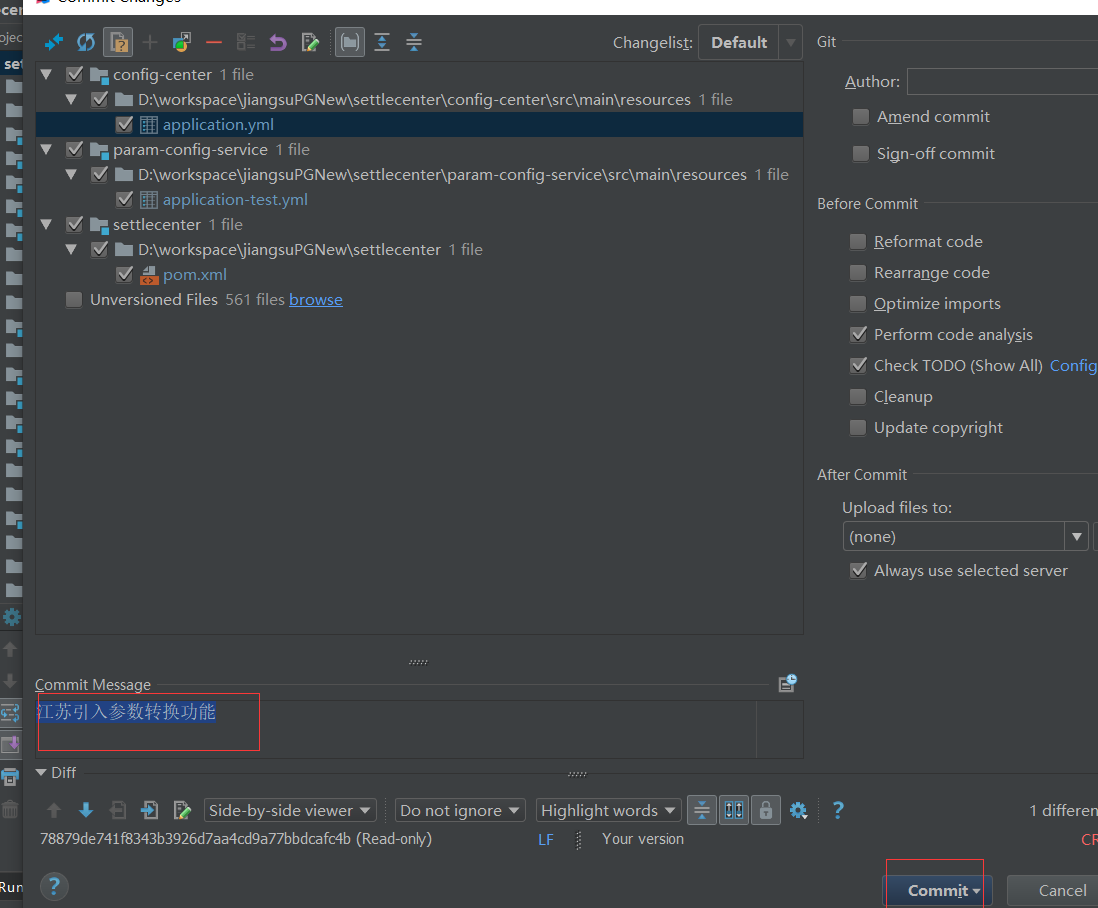




### 7.11.2在分支上开发，提交到分支

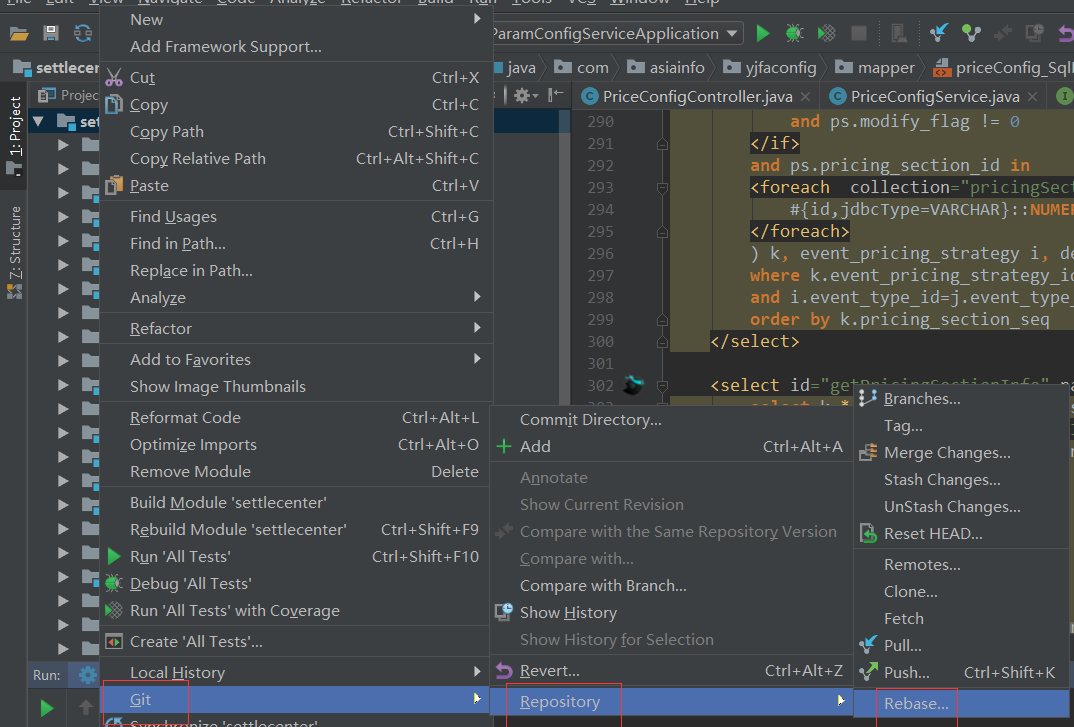
新建分支后，默认当前的开发就在分支上。直接在分支上进行开发提交，默认提交的也是提交到分支的。

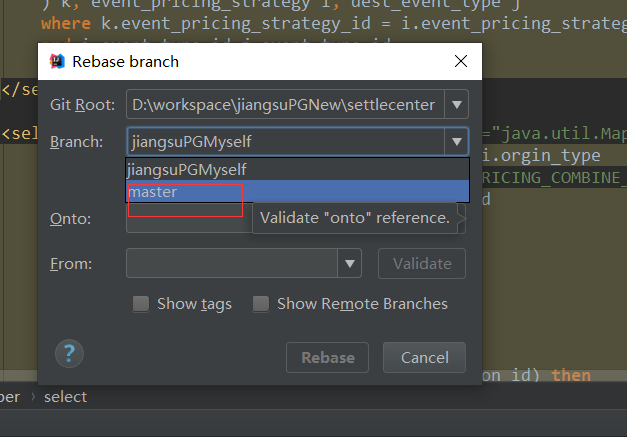


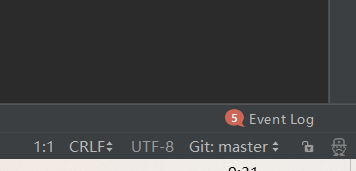


### 7.11.3切换分支

Git==>Repository==>Rebase 选择要切换的分支；切换完成后再IDEA的有右下角能看到master，说明已经切换到master分支了。

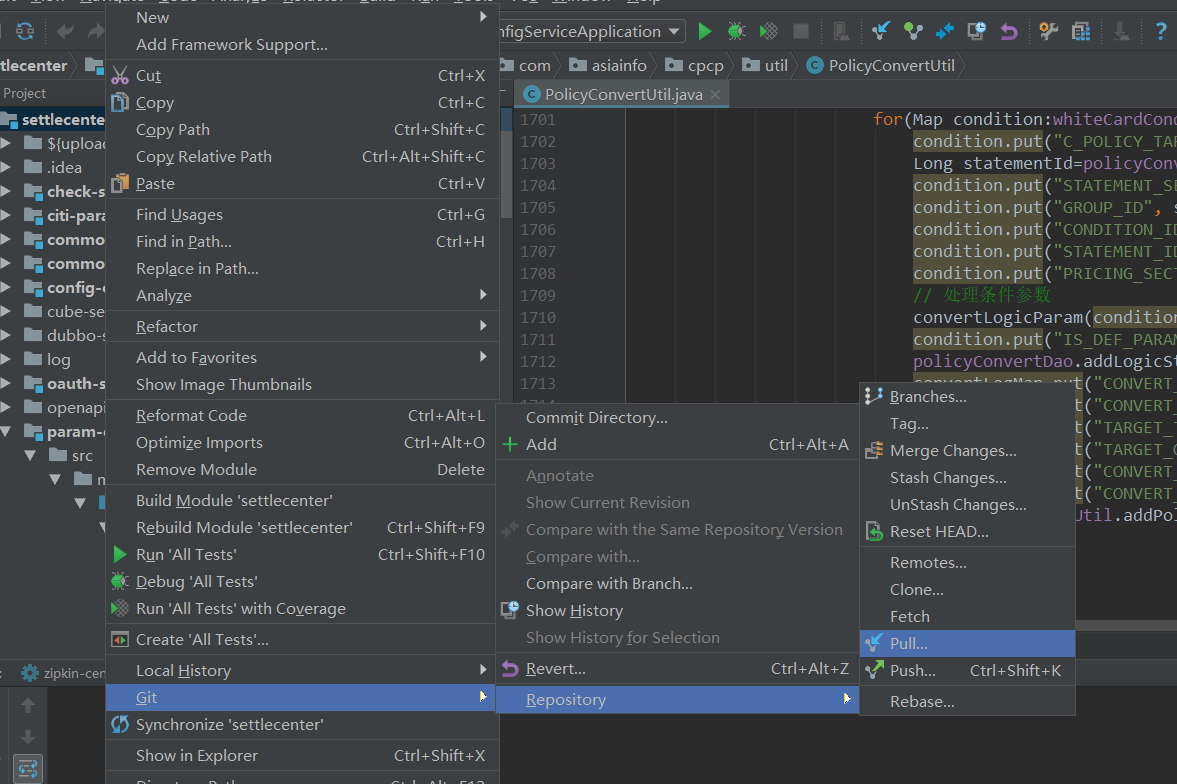




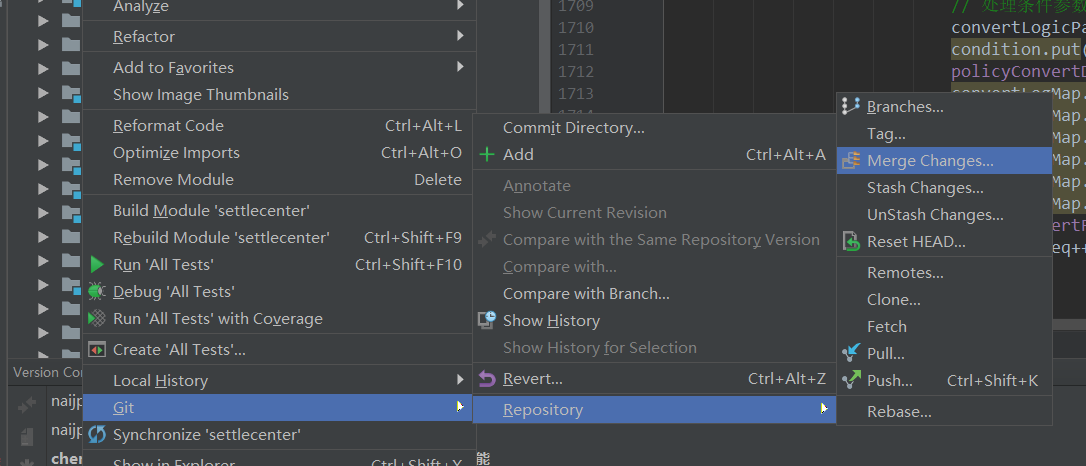


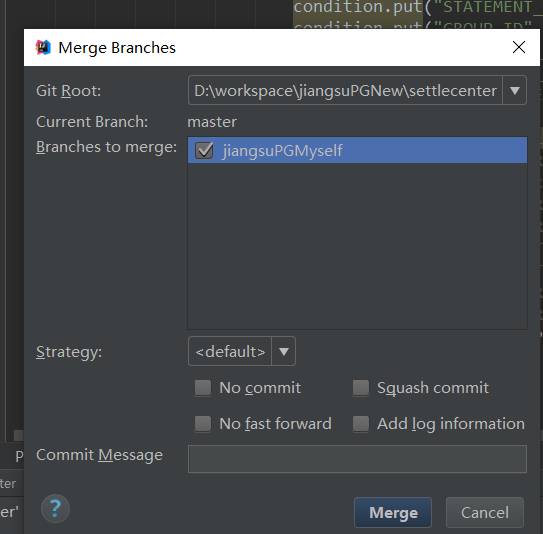
### 7.11.4合并分支

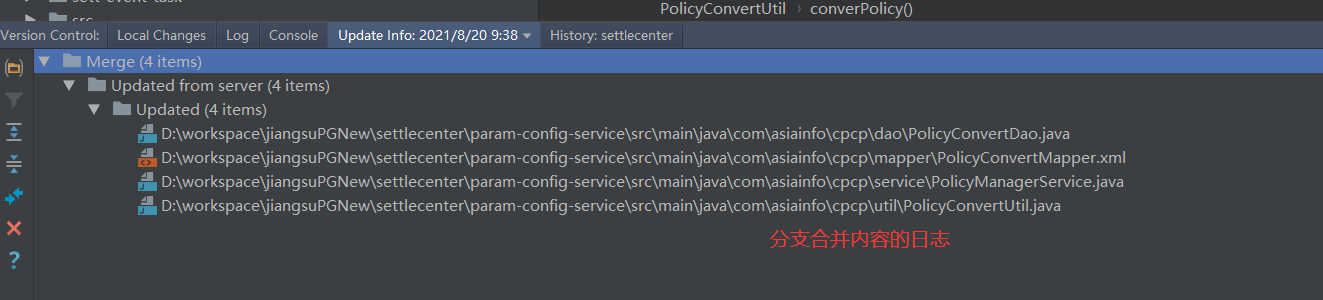
1. 合并分支之前注意点，当前版本一定切换到被合并的到分支进行操作，例如我们需要将新建分支代码合并到master上，需要将版本切换到master进行
2. 同步被合并到分支的代码，Git ==>Repository==>pull
3. 合并分支，Git==>Repository==>Merge Changes==>选择需要合并到主分支的上分支，点击Merge即完成分支的合并
4. 合并完分支如果有冲突则解决冲突，没有冲突可以push远程库了（前提是改分支的代码已经测试完成并且没有问题）

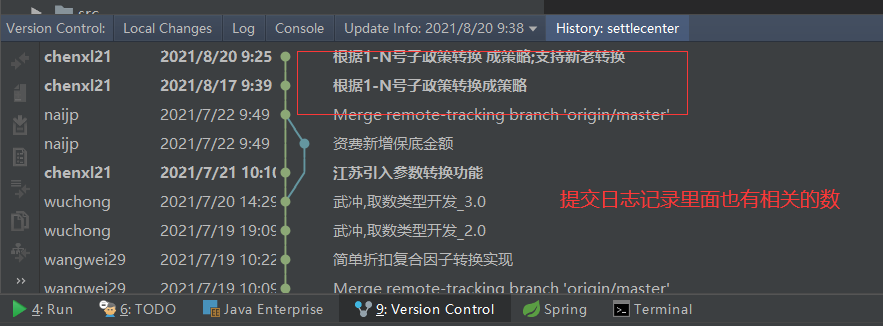


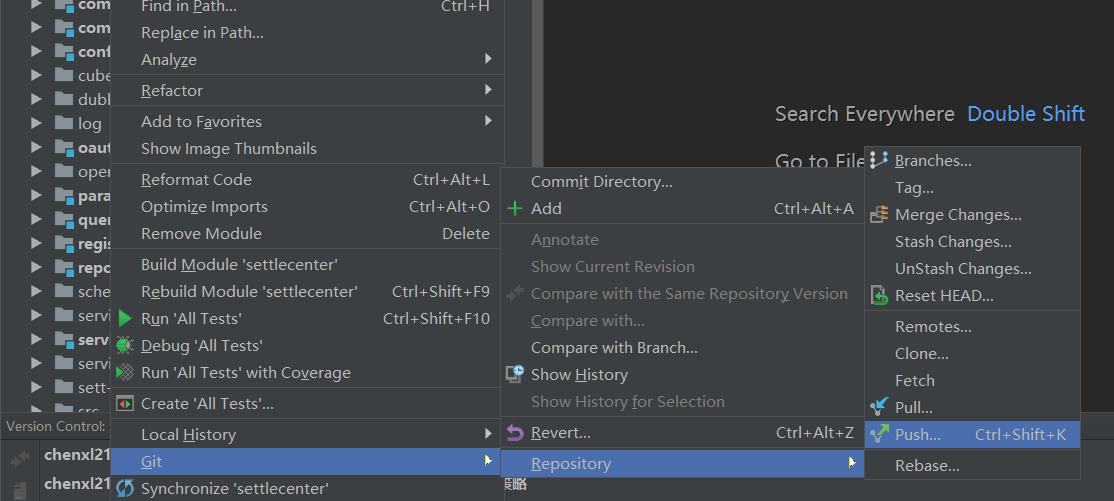


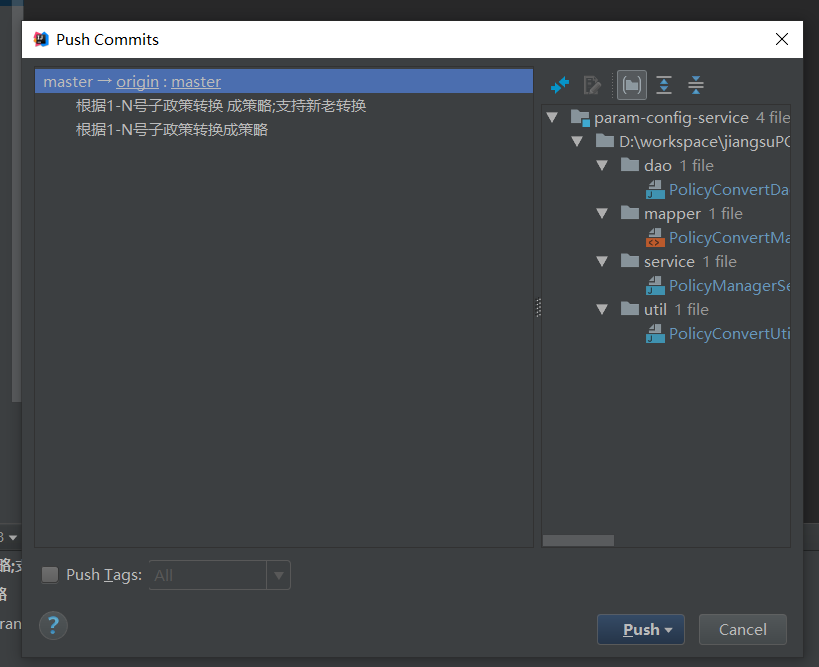












# IDEA集成GitHub

# 国内代码托管中心-码云

# 自建代码托管平台-GitLab