

1.Git介绍

1.1 版本控制(理解)

无论是代码编写，还是文档编写，我们都会遇到对文档内容反复修改的情况

-  1-毕业论文 (初版)
-  2-毕业论文 (完成版)
-  3-毕业论文 (修改版)
-  4-毕业论文 (最终版)
-  5-毕业论文 (最最终版)
-  6-毕业论文 (绝对不改版)
-  7-毕业论文 (打死不改版)

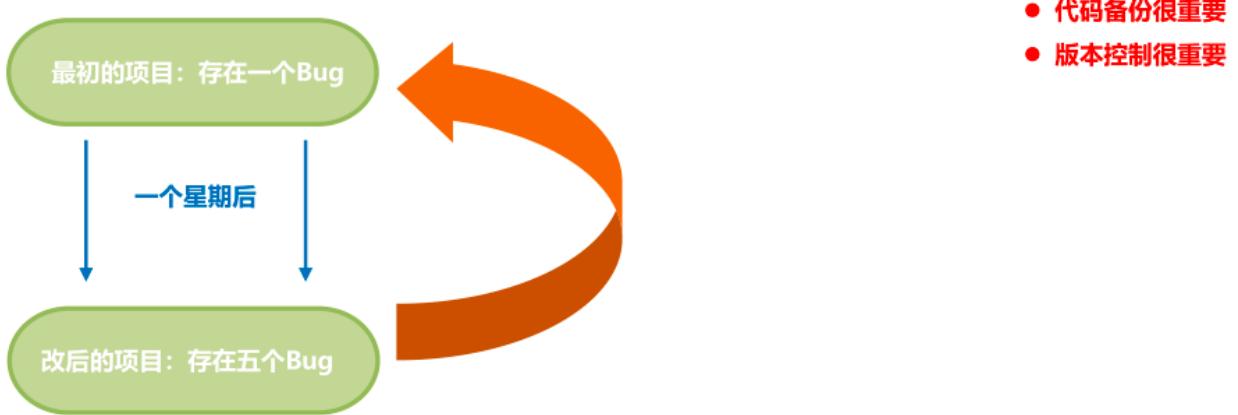
1.2 开发中存在的问题(理解)

- 程序员小明负责的模块就要完成了，就在即将提交发布之前的一瞬间，电脑突然蓝屏，硬盘光荣下岗！

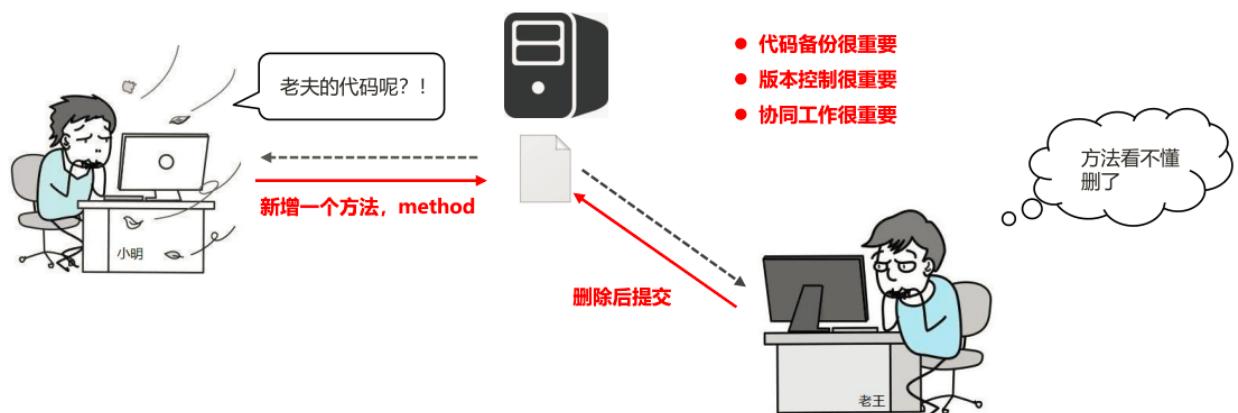
几个月来的努力付之东流



- 老王需要在项目中加入一个很复杂的功能，一边尝试，一边修改代码，就这样摸索了一个星期。可是这被改得面目全非的代码已经回不到从前了。



- 小明和老王先后从文件服务器上下载了同一个文件



- 因项目中Bug过多，导致项目进度拖延，项目经理老徐因此被骂，但不清楚Bug是手下哪一个程序员写的



- 开发中要解决的问题

- 代码备份
- 版本控制
- 协同工作
- 责任追溯

1.3SVN版本控制(理解)

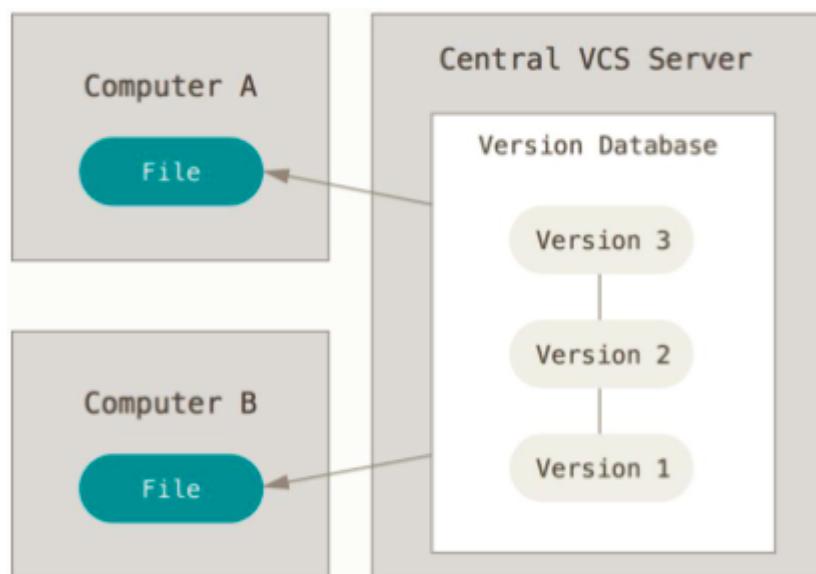
SVN是集中式版本控制系统，版本库是集中放在中央服务器的，而开发人员工作的时候，用的都是自己的电脑，所以首先要从中央服务器下载最新的版本，然后开发，开发完后，需要把自己开发的代码提交到中央服务器。

- 服务器单点故障

将会导致所有人员无法工作

- 而服务器硬盘损坏

这意味着，你可能失去了该项目的所有历史记录，这是毁灭性的。

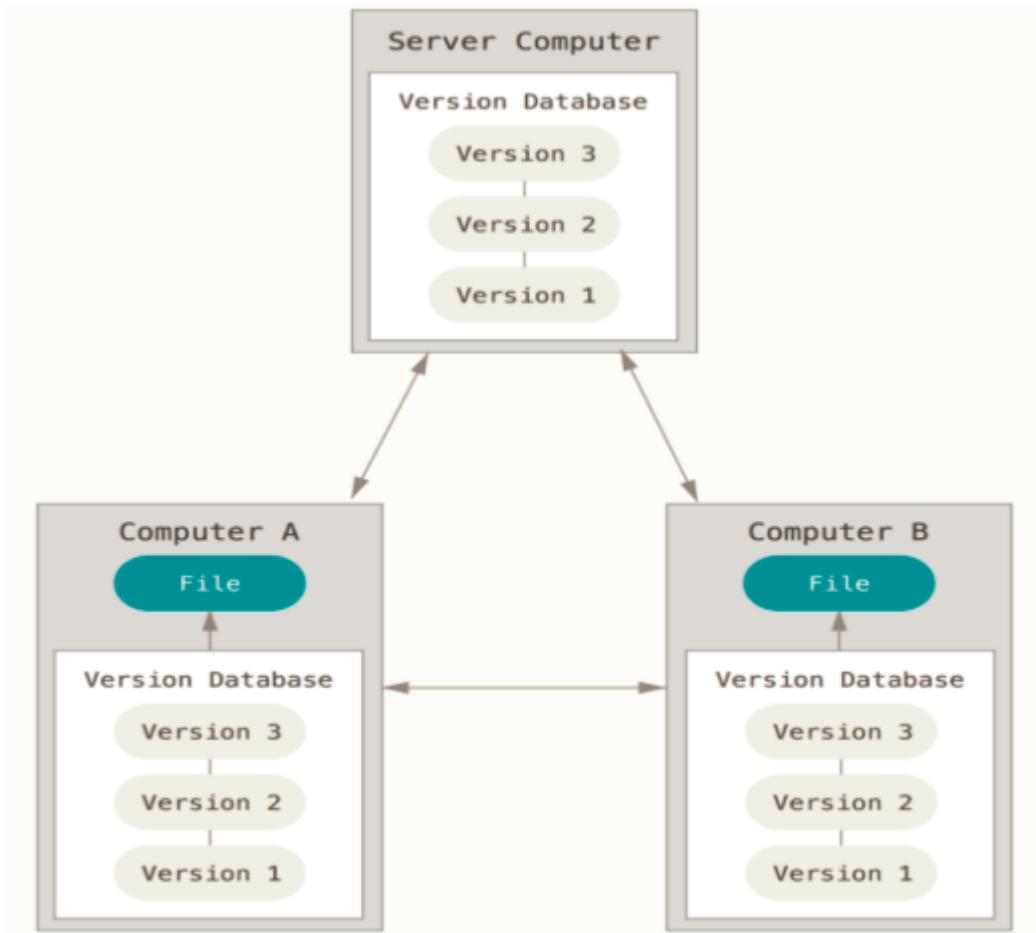


1.4Git版本控制(理解)

Git是在2005年，Linux系统的创建者Linus Torvalds,为了帮助全球的开发者，维护Linux系统内核的开发而开发了自己的开源分布式版本控制工具,分为两种类型的仓库：本地仓库和远程仓库。

- 每一个客户端都保存了完整的历史记录

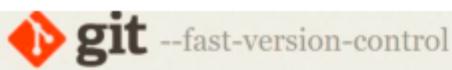
服务器的故障，都可以通过客户端的记录得以恢复。



2.Git下载和安装

2.1 Git的下载(应用)

官网下载地址: <https://git-scm.com/downloads>



Search entire site...

About
Documentation
Downloads
 GUI Clients
 Logos
Community

The entire [Pro Git book](#) written by Scott Chacon and Ben Straub is available to [read online for free](#). Dead tree versions are available on [Amazon.com](#).

Downloads

Mac OS X Windows

Linux/Unix

Older releases are available and the [Git source repository](#) is on GitHub.



GUI Clients

Git comes with built-in GUI tools (`git-gui`, `gitk`), but there are several third-party tools for users looking for a platform-specific experience.

[View GUI Clients →](#)

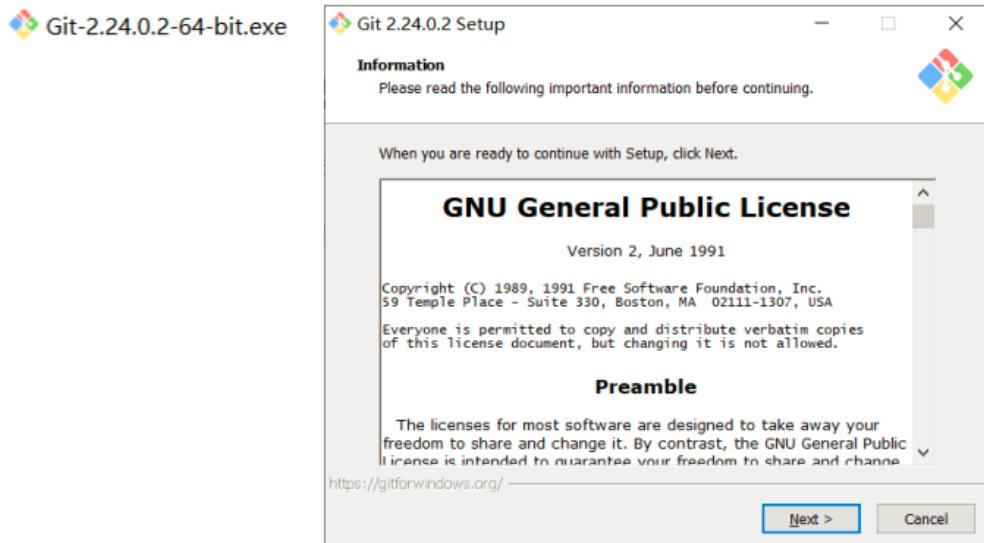
Logos

Various Git logos in PNG (bitmap) and EPS (vector) formats are available for use in online and print projects.

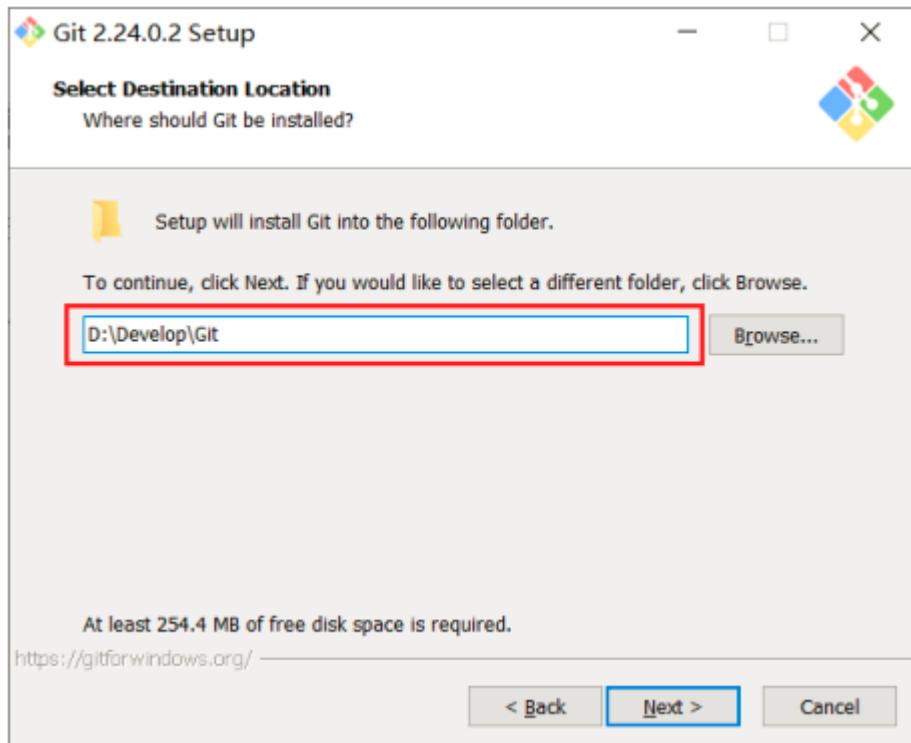
[View Logos →](#)

2.2 Git的安装(应用)

1. 双击安装包，进入安装向导界面

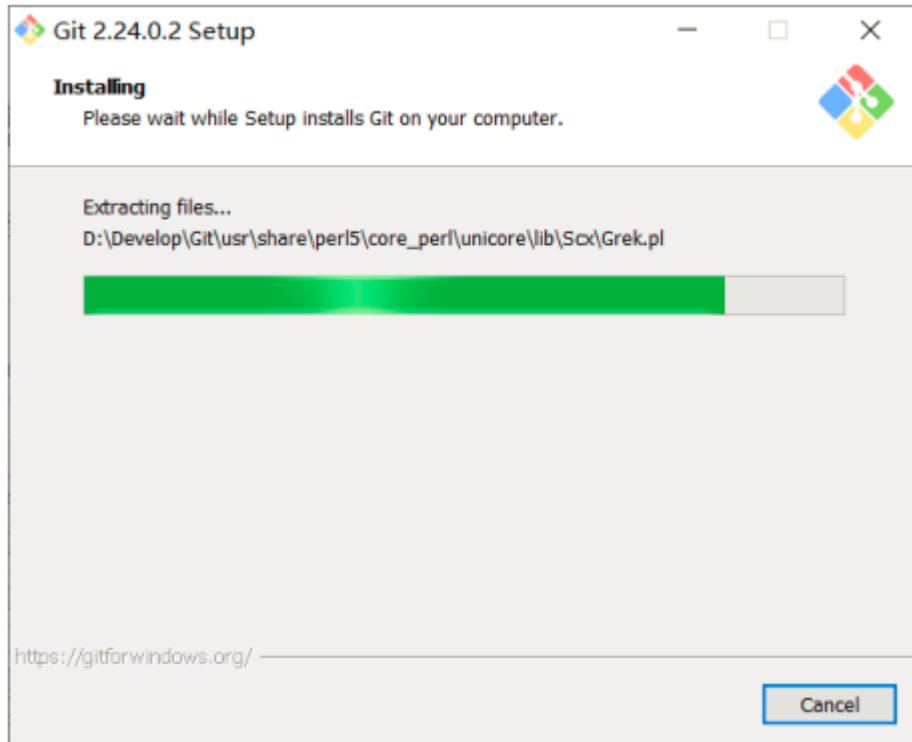


2. 指定安装目录

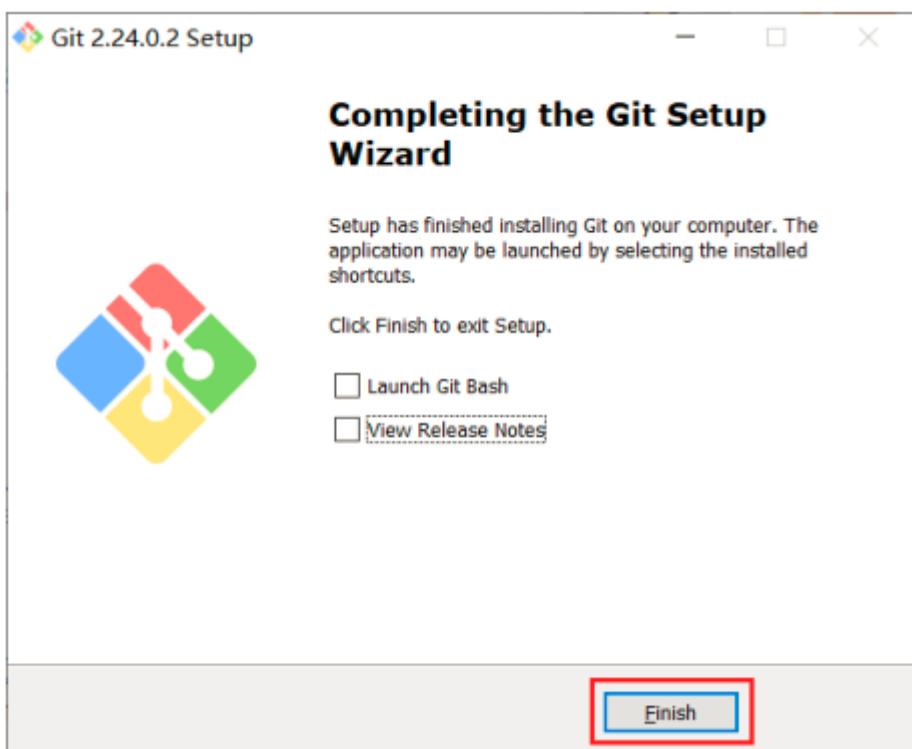


3. 一路next下一步

4. 等待安装



5. 安装完成



6. 安装完成后在电脑桌面（也可以是其他目录）点击右键，如果能够看到如下两个菜单则说明Git安装成功。



Git GUI: Git提供的图形界面工具

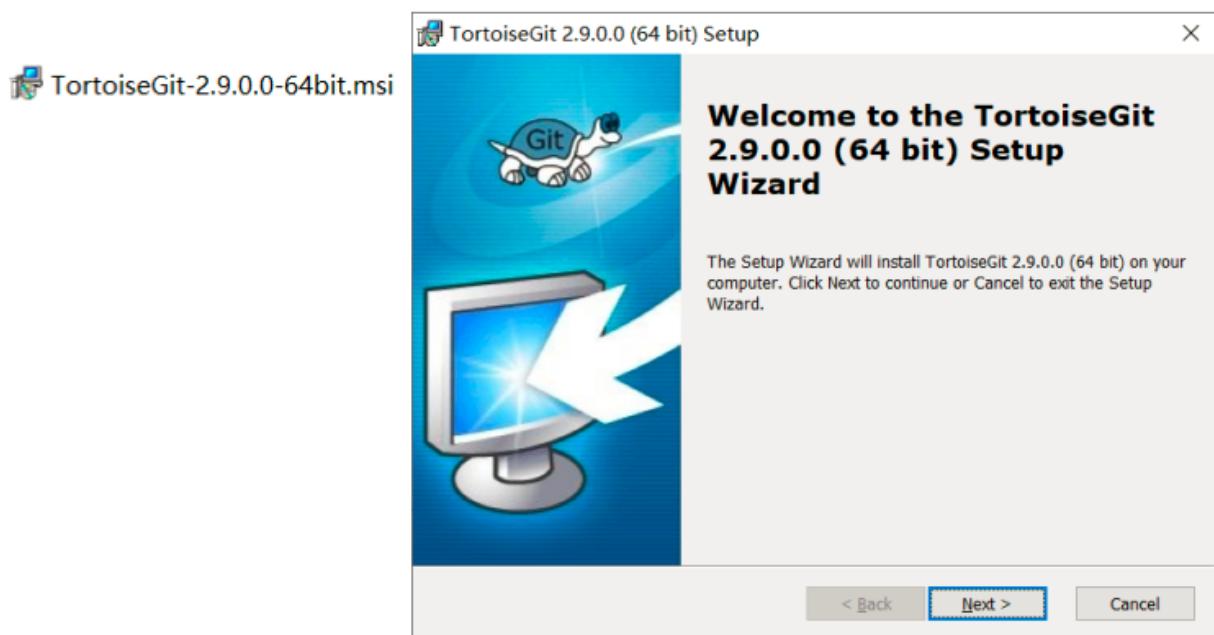
Git Bash: Git提供的命令行工具

7. 运行Git命令客户端，使用git --version 命令，可以查看git版本

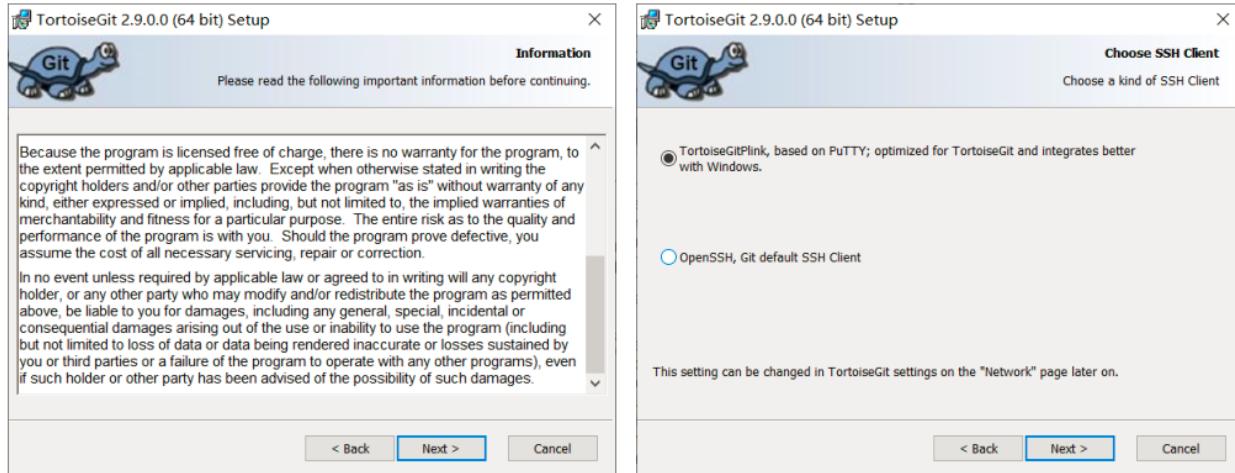
```
MINGW64:/c/Users/ /Desktop
admin@DESKTOP-LOIANGVO MINGW64 ~/Desktop
$ git --version
git version 2.24.0.windows.2
admin@DESKTOP-LOIANGVO MINGW64 ~/Desktop
$
```

2.3 TortoiseGit的安装(应用)

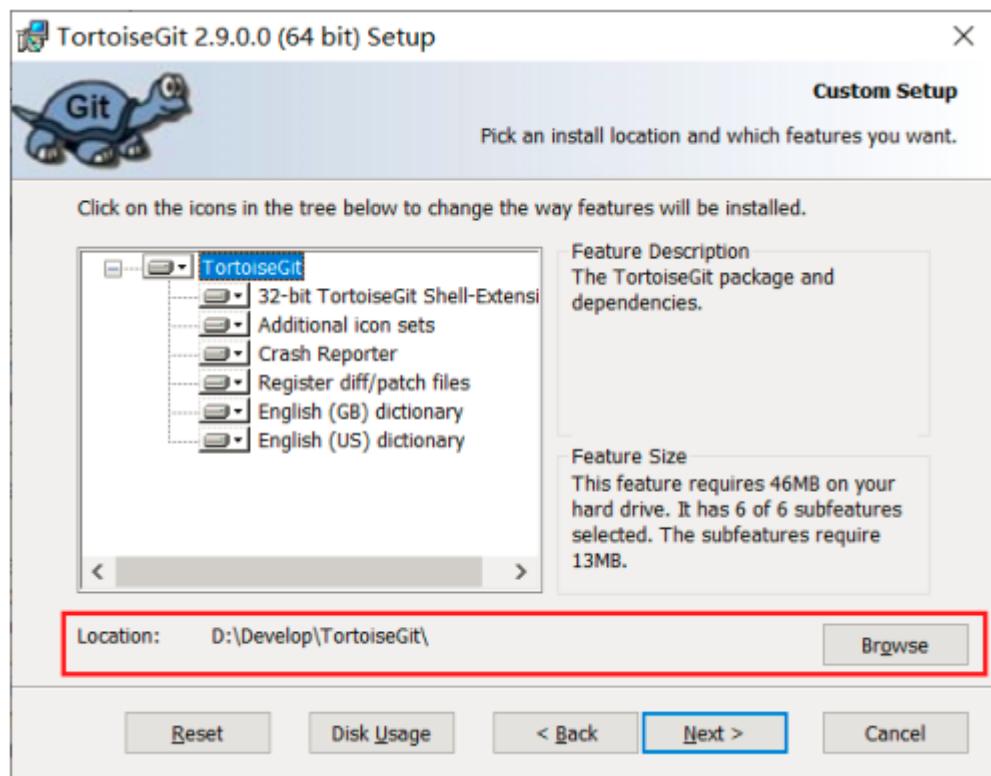
1. 双击安装包，进入安装向导界面



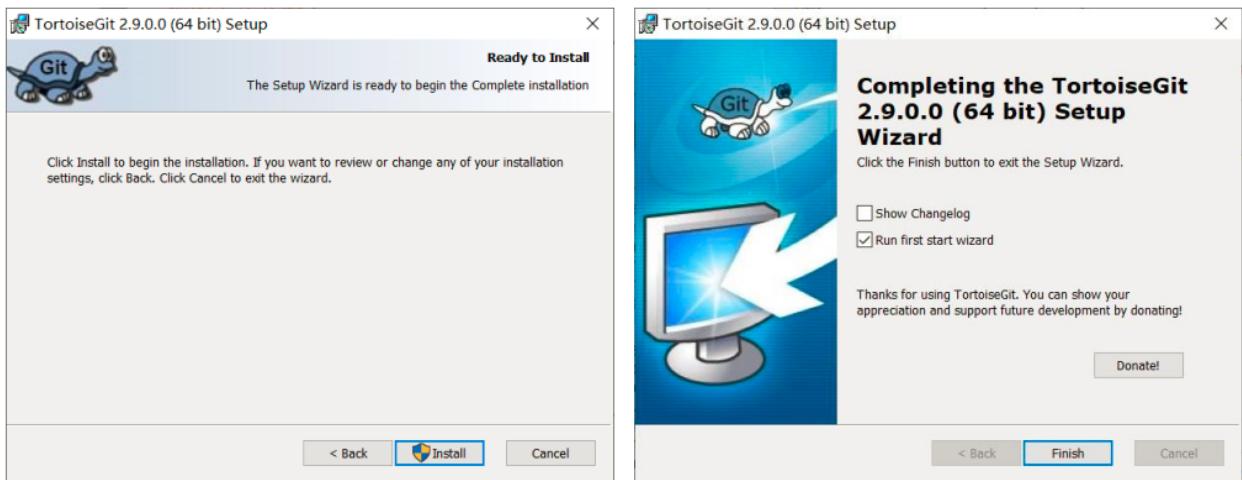
2. 一路next下一步



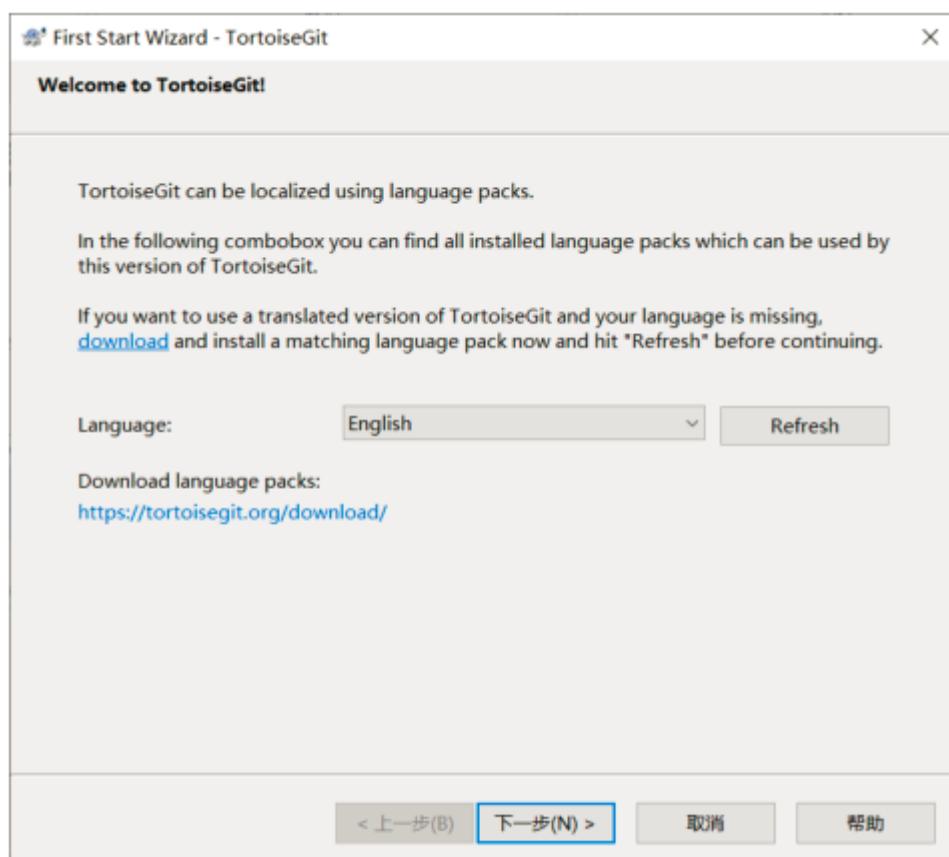
3. 指定安装目录

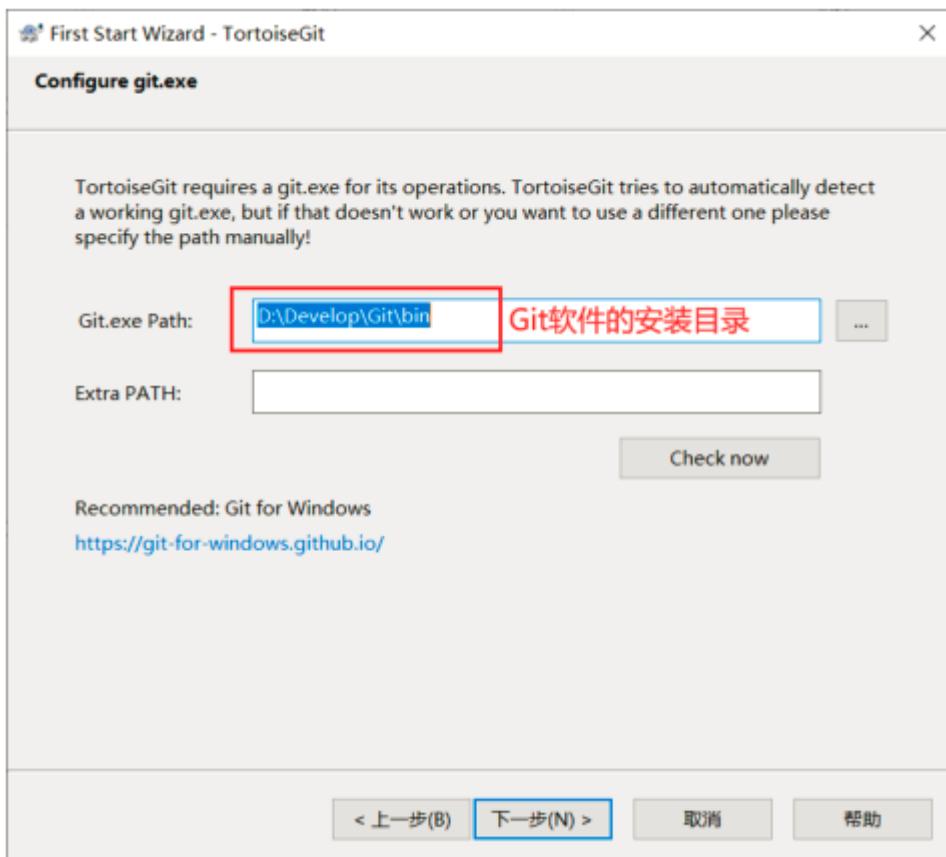
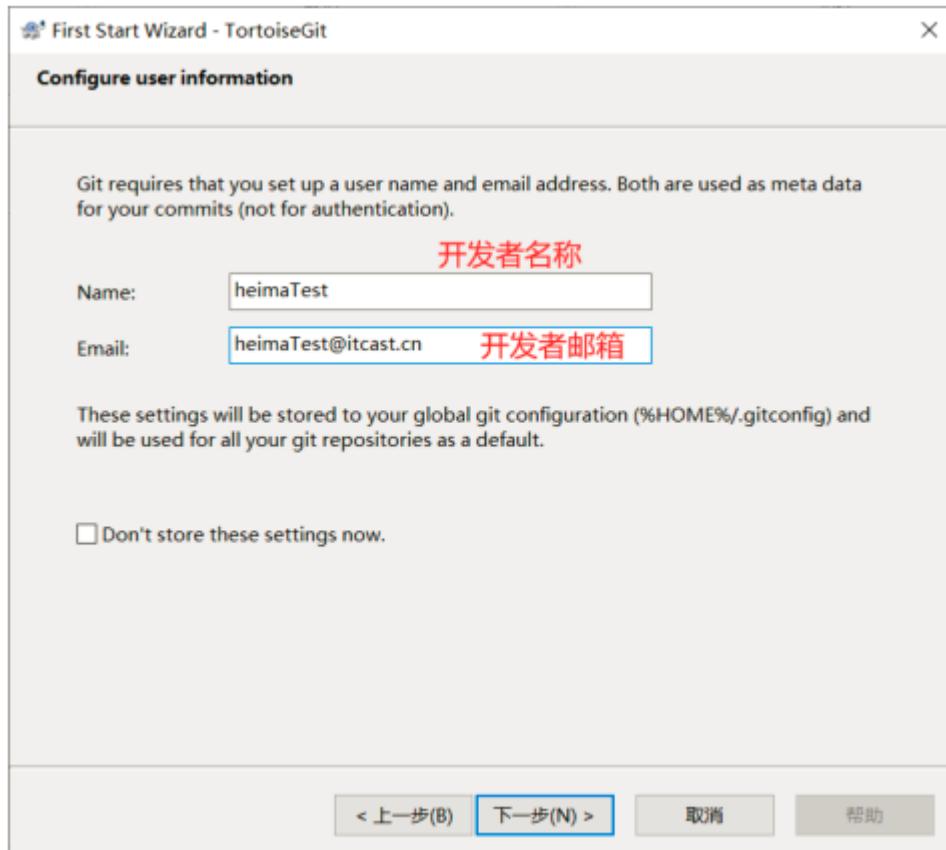


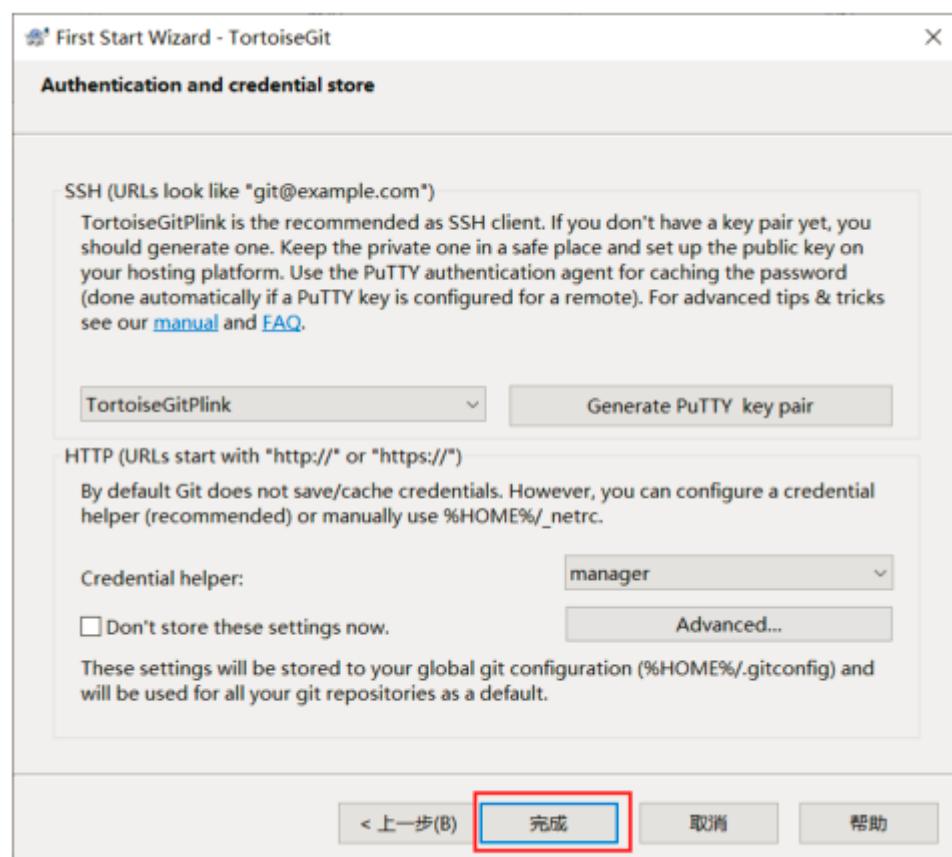
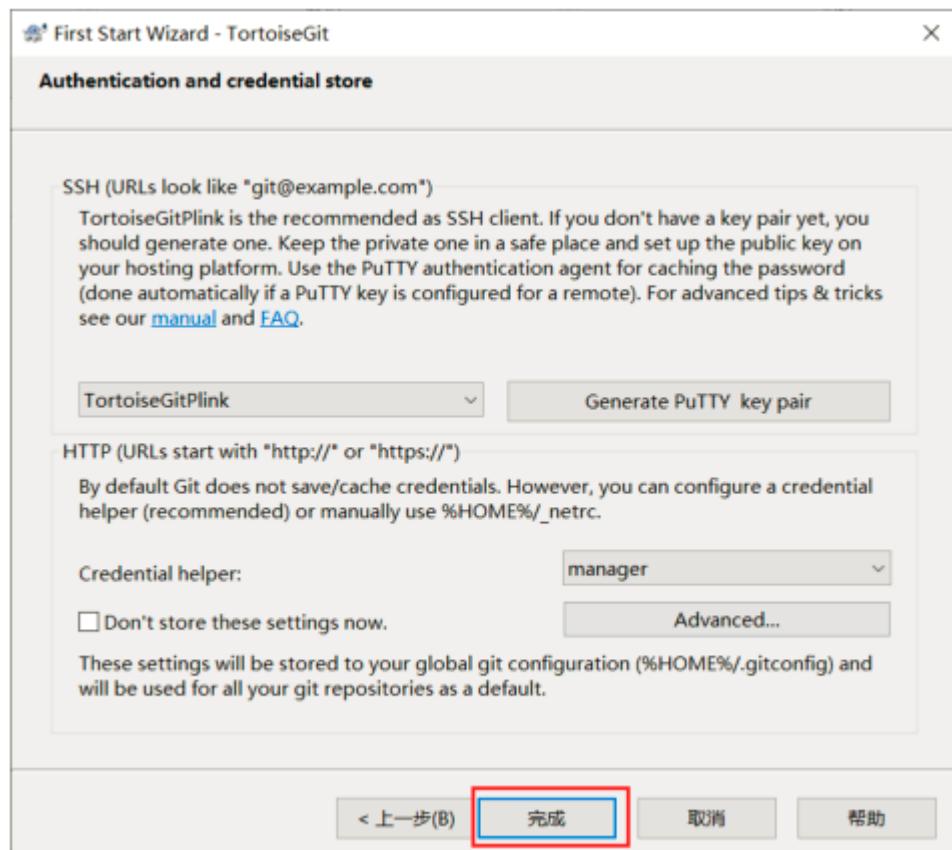
4. 安装

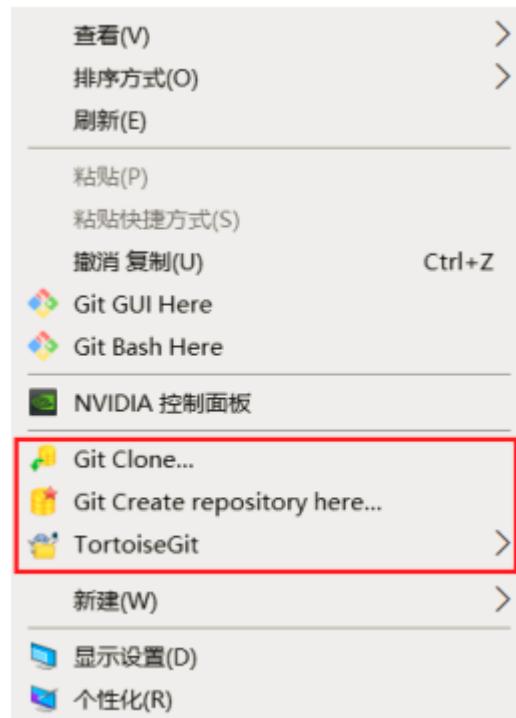


5. 配置

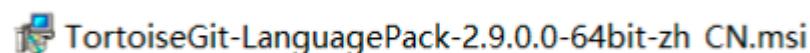






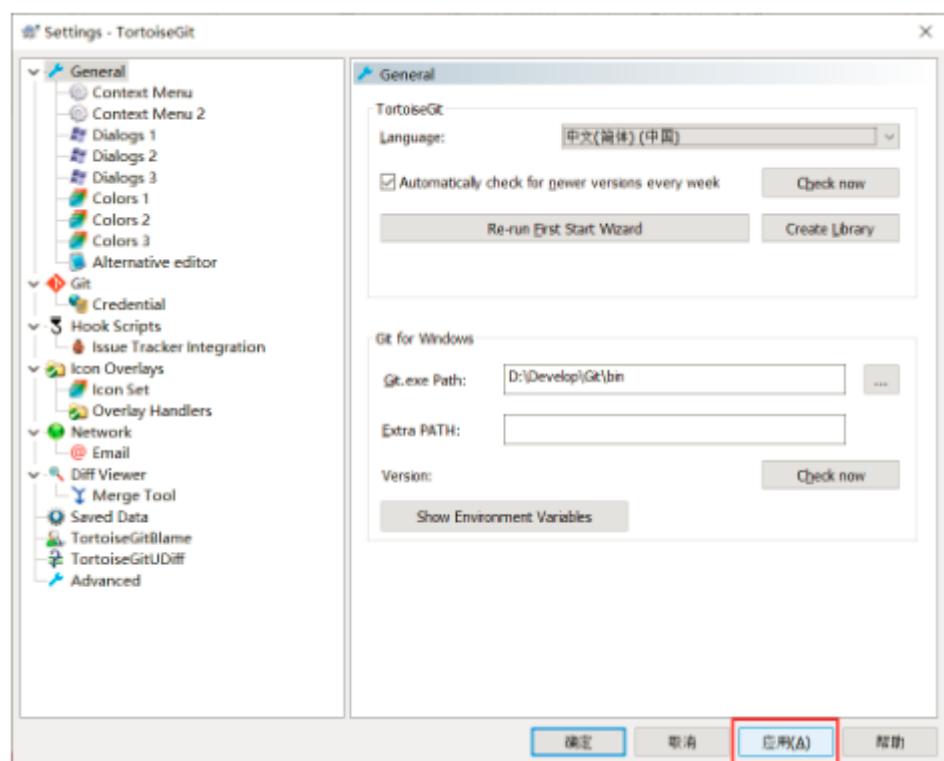
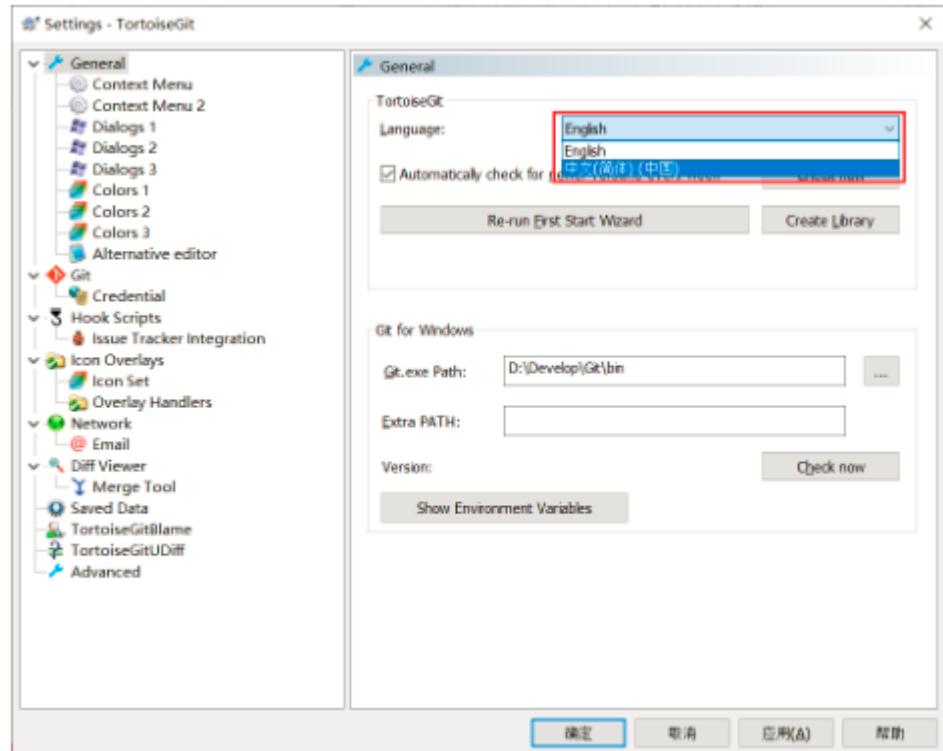


6. 安装TortoiseGit中文语言包,一路next即可



7. 配置TortoiseGit中文语言







3.Git操作入门

3.1 Git基本工作流程(理解)

本地仓库



3.2 Git命令行操作(应用)

- git常用命令

命令	作用
git init	初始化，创建 git 仓库
git status	查看 git 状态（文件是否进行了添加、提交操作）
git add 文件名	添加，将指定文件添加到暂存区
git commit -m '提交信息'	提交，将暂存区文件提交到历史仓库
git log	查看日志（git 提交的历史日志）

- 操作步骤

- 创建工作目录、初始化本地 git 仓库

共享 查看

计算机 > Work (D:) > my_project

名称

.git

MINGW64:/d/my_project

```
admin@LAPTOP-LO4AUSV0 MINGW64 /d/my_project
$ git init
Initialized empty Git repository in D:/my_project/.git/
admin@LAPTOP-LO4AUSV0 MINGW64 /d/my_project (master)
$
```

- 新建一个 test.txt 文件（暂不执行添加操作）

- 使用 status 命令，查看状态

> 计算机 > Work (D:) > my_project

名称
.git
test.txt

```
MINGW64:/d/my_project
admin@LAPTOP-L04AUSV0 MINGW64 /d/my_project
$ git init
Initialized empty Git repository in D:/my_project/.git/
admin@LAPTOP-L04AUSV0 MINGW64 /d/my_project (master)
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    [test.txt]

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

admin@LAPTOP-L04AUSV0 MINGW64 /d/my_project (master)
$ |
```

4. 使用 add 命令添加，并查看状态

> 计算机 > Work (D:) > my_project

名称
.git
test.txt

若没有出现 (加号) 图标，请重新电脑

名称
.git
test.txt

```
MINGW64:/d/my_project
admin@LAPTOP-L04AUSV0 MINGW64 /d/my_project (master)
$ git add test.txt
admin@LAPTOP-L04AUSV0 MINGW64 /d/my_project (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
    new file:   test.txt

admin@LAPTOP-L04AUSV0 MINGW64 /d/my_project (master)
$ |
```

5. 使用 commit 命令，提交到本地历史仓库

> 计算机 > Work (D:) > my_project

名称



.git



test.txt

MINGW64:/d/my_project

```
admin@LAPTOP-L04AUSV0 MINGW64 /d/my_project (master)
$ git commit -m 'commit first file test.txt'
[master (root-commit) 5dd9a30] commit first file test.txt
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 test.txt

admin@LAPTOP-L04AUSV0 MINGW64 /d/my_project (master)
$
```

git commit **-m** ‘提交时，携带的描述信息’

6. 使用 log 命令，查看日志

MINGW64:/d/my_project

```
admin@LAPTOP-L04AUSV0 MINGW64 /d/my_project (master)
$ git log
commit 5dd9a300e7ce021e2e595d49793cbf1a83eff650 (HEAD -> master)
Author: heimaTest <heimaTest@itcast.cn>
Date:   Sat Jan 11 15:59:23 2020 +0800

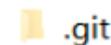
    commit first file test.txt

admin@LAPTOP-L04AUSV0 MINGW64 /d/my_project (master)
$
```

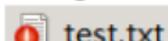
7. 修改 test.txt 文件

> 计算机 > Work (D:) > my_project

名称



.git



test.txt

test.txt - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

update count=1

8. 添加并提交，查看日志

> 计算机 > Work (D:) > my_project >

名称
.git
test.txt

MINGW64:/d/my_project

```
admin@LAPTOP-L04AUSV0 MINGW64 /d/my_project (master)
$ git add test.txt

admin@LAPTOP-L04AUSV0 MINGW64 /d/my_project (master)
$ git commit -m 'commit second file test.txt'
[master 3471d47] commit second file test.txt
1 file changed, 1 insertion(+)

admin@LAPTOP-L04AUSV0 MINGW64 /d/my_project (master)
$ git log
commit 3471d4760fc40be03b6172b975385c25144e120b (HEAD -> master)
Author: heimatest <heimatest@itcast.cn>
Date:   Sat Jan 11 16:15:04 2020 +0800

    commit second file test.txt

commit db8d3831d08bd237e24f5f2408e41f923fa6c996
Author: heimatest <heimatest@itcast.cn>
Date:   Sat Jan 11 16:13:29 2020 +0800

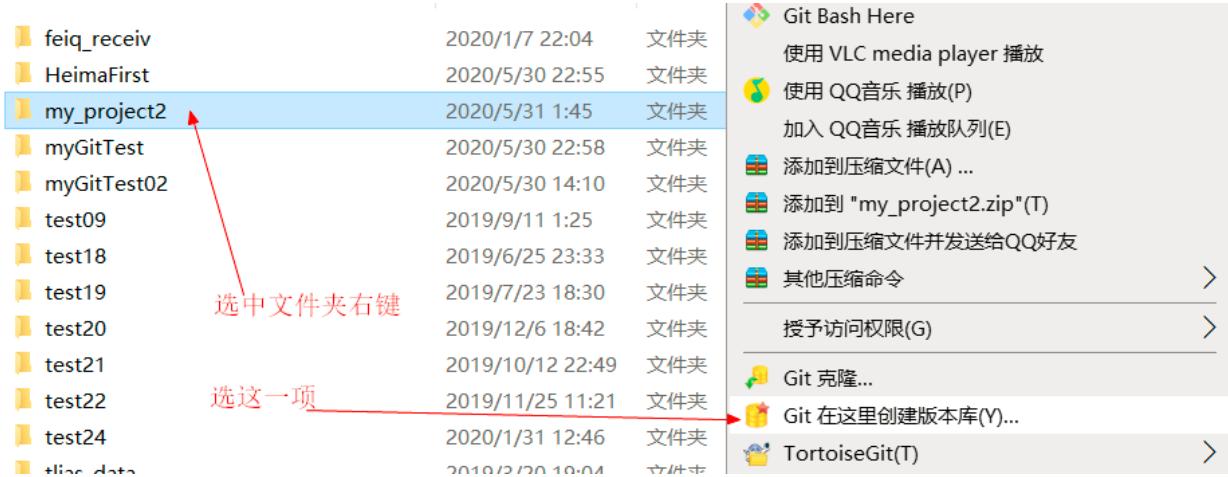
    commit first file test.txt

admin@LAPTOP-L04AUSV0 MINGW64 /d/my_project (master)
$
```

注意：白色框中的内容

3.3 Git图形化工具操作(理解)

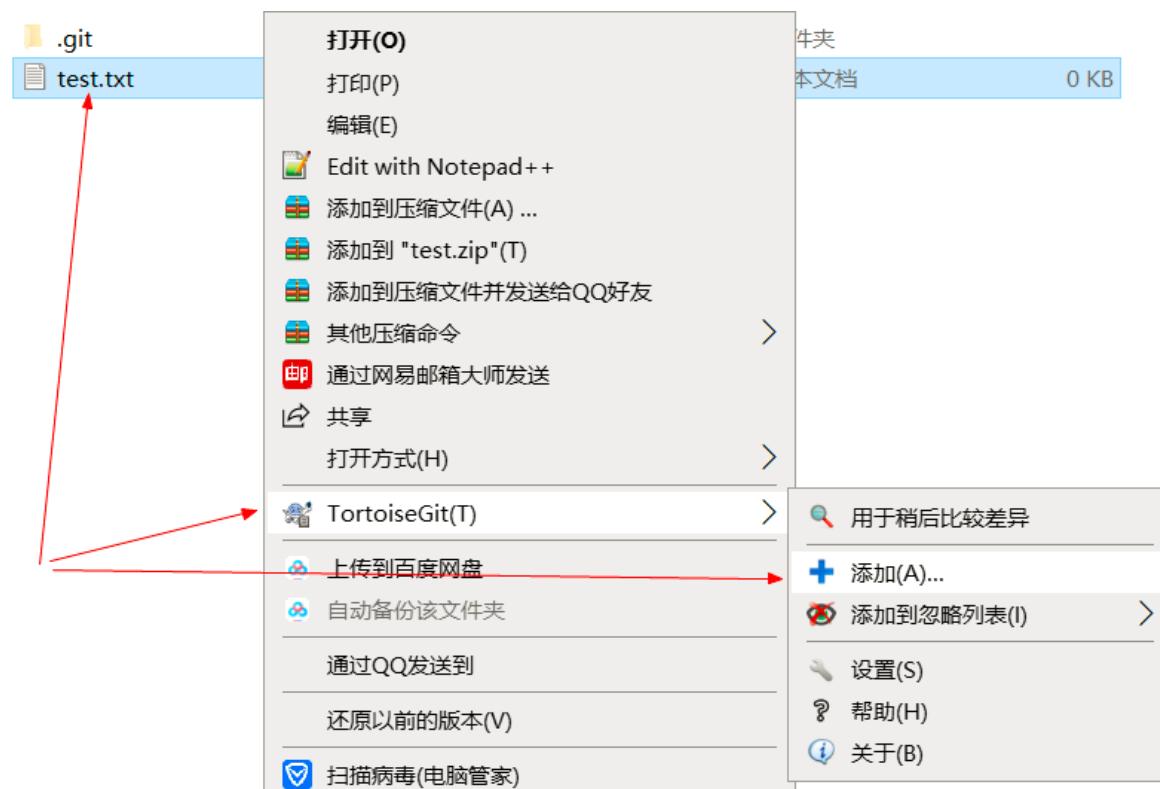
1. 创建工作目录、初始化本地 git 仓库





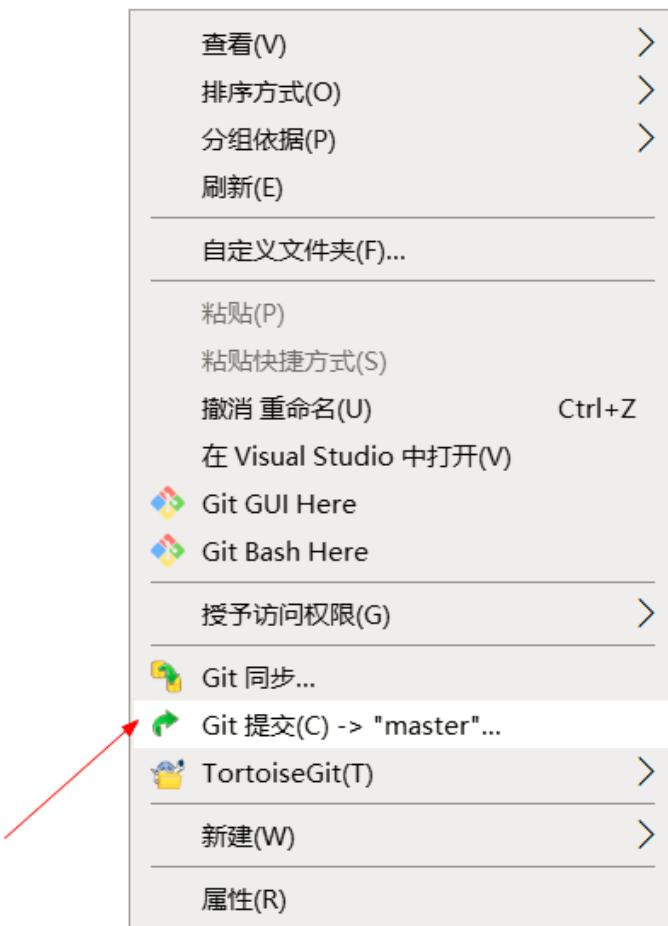
2. 新建一个 test.txt 文件 (暂不执行添加操作)

3. 选中文件右键，选择TortoiseGit，之后选择添加

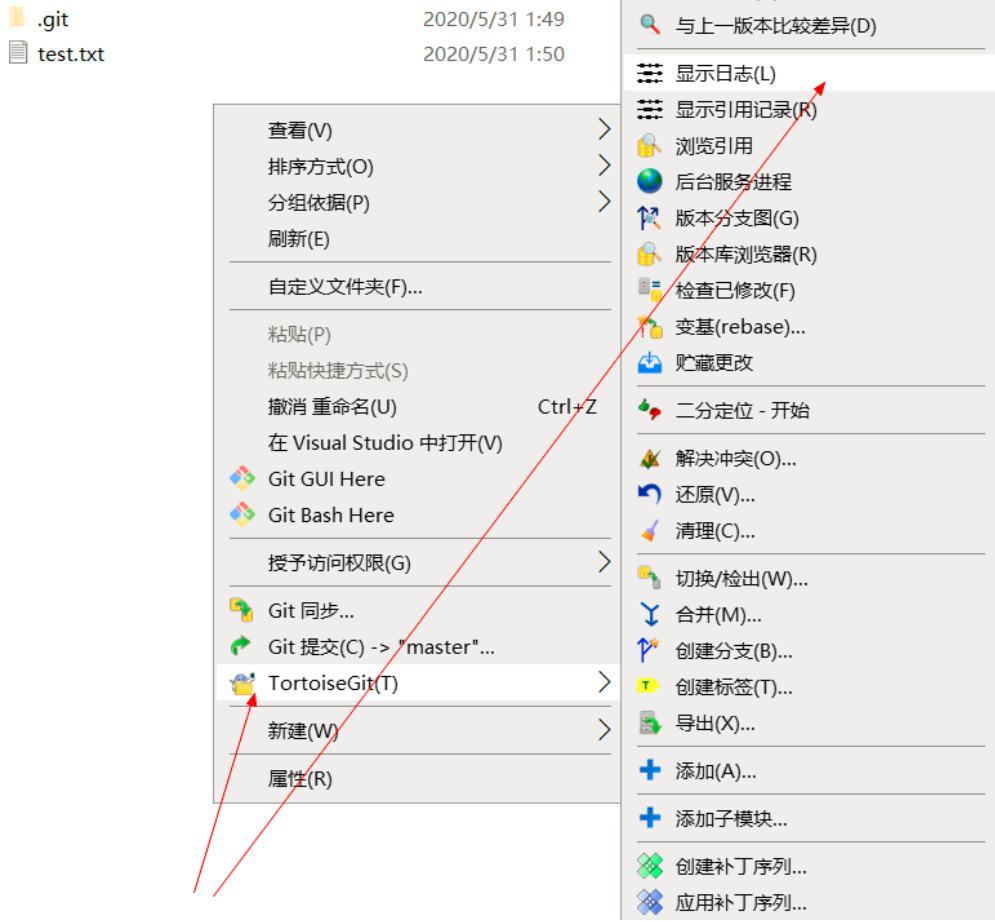


4. 空白处右键, Git提交, 提交到本地历史仓库

📁 .git	2020/5/31 1:49	文件夹
📄 test.txt	2020/5/31 1:50	文本文档



5. 空白处右键,TortoiseGit,显示日志,可以产看日志信息

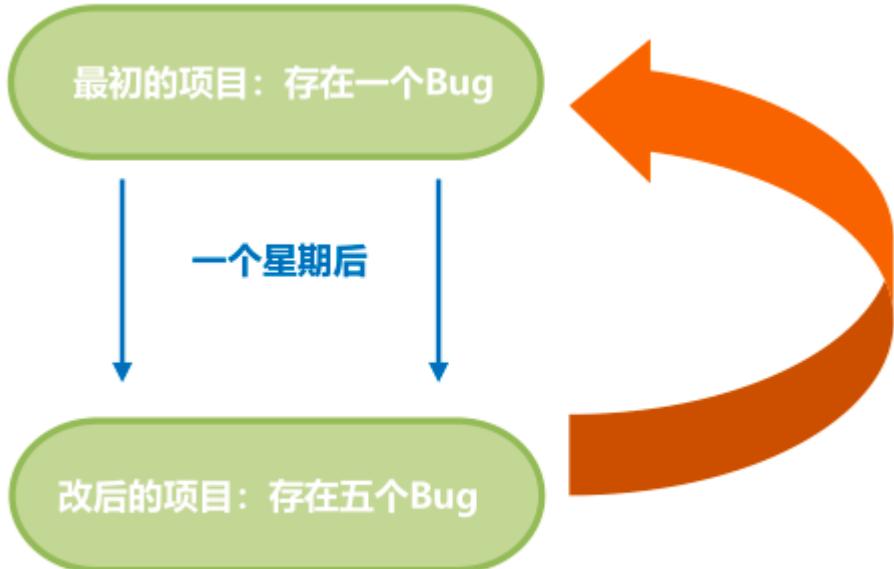


6. 修改 test.txt 文件

7. 添加并提交，查看日志

4.Git版本管理

4.1历史版本切换(理解)

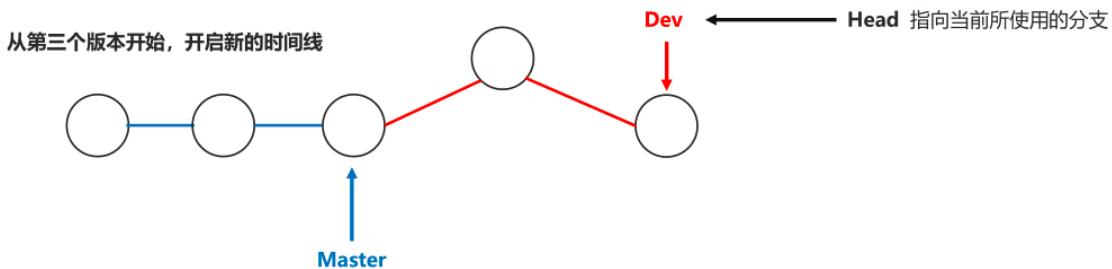


- 准备动作
 1. 查看 my_project 的 log 日志
git reflog : 可以查看所有分支的所有操作记录 (包括已经被删除的 commit 记录的操作)
 2. 增加一次新的修改记录
- 需求: 将代码切换到第二次修改的版本
指令: git reset --hard 版本唯一索引值

4.2 分支管理介绍(理解)

- 分支
 - 由每次提交的代码，串成的一条时间线
 - 使用分支意味着你可以把你的工作从开发主线上分离开来,以免影响开发主线
- 分支的使用场景
 1. 周期较长的模块开发
假设你准备开发一个新功能，但是需要一个月才能完成
第一周写了20%的代码，突然发现原来已经写好的功能出现了一个严重的Bug
那现在就需要放下手中的新功能，去修复Bug
但这20%的代码不能舍弃，并且也担心丢失，这就需要开启一个新的版本控制。
 2. 尝试性的模块开发
业务人员给我们提出了一个需求，经过我们的思考和分析
该需求应该可以使用技术手段进行实现。
但是我们还不敢确定，我们就可以去创建一个分支基于分支进行尝试性开发。

- 分支工作流程
 - Master: 指向提交的代码版本
 - Header: 指向当前所使用的分支



4.3 分支管理操作(应用)

- 创建和切换

创建命令: git branch 分支名 切换命令: git checkout 分支名

- 新分支添加文件

查看文件命令: ls

总结: 不同分支之间的关系是平行的关系, 不会相互影响

- 合并分支

合并命令: git merge 分支名

- 删除分支

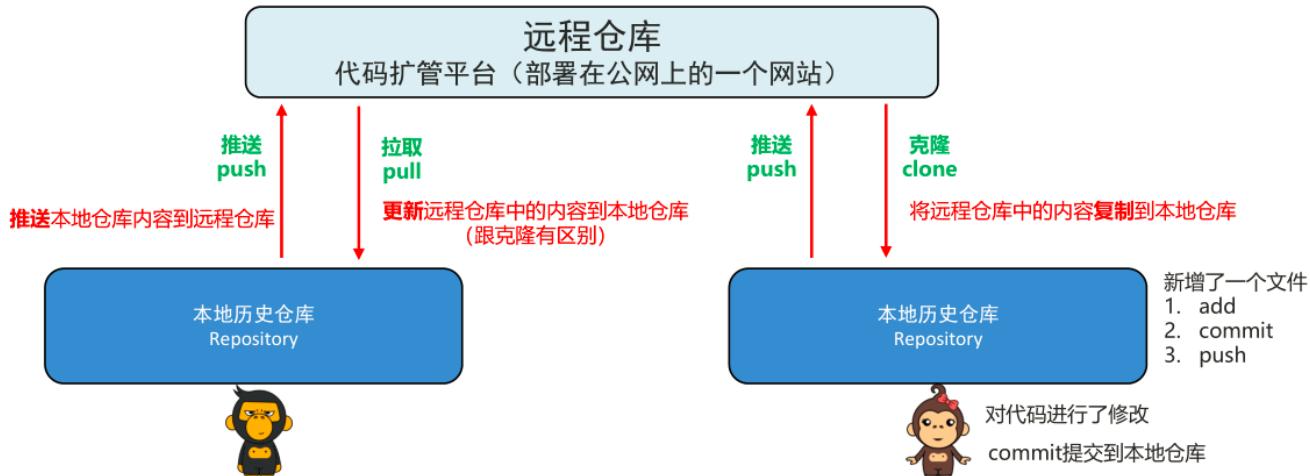
删除命令: git branch -d 分支名

- 查看分支列表

查看命令: git branch

5. 远程仓库

5.1 远程仓库工作流程(理解)



5.2 远程仓库平台介绍(理解)

- GitHub

域名: <https://github.com> 介绍: GitHub是全球最大的开源项目托管平台, 俗称大型程序员社区化交友网站

各类好玩有趣的开源项目, 只有想不到, 没有找不到。

- 码云

域名: <https://gitee.com> 介绍: 码云是全国最大的开源项目托管平台, 良心平台, 速度快, 提供免费私有库

5.3 码云的注册(应用)





黑马 Java

开源软件 企业版 **高校版** 博客 我的码云

搜索或跳转...

黑马 Java

个人主页 动态

仓库 0
Pull Requests 0
任务 0
代码片段 0

无数据

所有动态

您的帐号尚未绑定邮箱。
点此 添加绑定

绑定邮箱后，通过命令
`git config user.email` 配置绑定邮箱。提交代码可累计在平台上的贡献度。

Git 新手？
点此查看《初次运行 Git 前的配置》

推荐关注

仓库

还没有仓库，立即创建 或从 Github 导入

推荐仓库 基于当前流行仓库

5.4 先有本地项目，远程为空(应用)

- 步骤

1. 创建本地仓库
2. 创建或修改文件，添加 (add) 文件到暂存区，提交 (commit) 到本地仓库
3. 创建远程仓库

4. 推送到远程仓库

- 创建远程仓库



新建仓库

仓库名称 hello-git

归属 黑马Java / 路径 hello-git

仓库地址: https://gitee.com/black_horse_java/hello-git

仓库介绍 非必填
用简短的语言来描述一下吧

是否开源 公开

任何人都可以访问该仓库的代码和其他任何形式的资源

选择语言 Java | 添加 .gitignore | 添加开源许可证

使用 README 文件初始化这个仓库

使用 ISSUE 模板文件初始化这个仓库 ①

使用 Pull Request 模板文件初始化这个仓库 ①

选择分支模型 (仓库初始化后将根据所选分支模型创建分支)
单分支模型 (只创建 master 分支)

导入已有仓库

创建

- 生成SSH公钥

- 推送代码之前，需要先配置SSH公钥

```
$ git push
remote: You do not have permission to push to the repository via HTTPS
fatal: Authentication failed for 'https://gitee.com/y****@gitee.com/repo1/'
```

- 生成SSH公钥步骤

- 设置Git账户

- git config user.name (查看git账户)
- git config user.email (查看git邮箱)
- git config --global user.name “账户名” (设置全局账户名)
- git config --global user.email “邮箱” (设置全局邮箱)
- cd ~/.ssh (查看是否生成过SSH公钥)

```
MINGW64:/d/my_project
$ git config user.name
heimaTest          查看用户名和邮箱
$ git config user.email
heimaTest@itcast.cn

$ git config --global user.name "heimaTest"
$ git config --global user.email "heimaTest@itcast.cn" --global: 表示这台机器上所有的Git仓库都会使用这个配置

$ cd ~/.ssh
bash: cd: /c/Users/haoys/.ssh: No such file or directory 如果看到这句话代表没有生成过SSH
admin@LAPTOP-L04AUSV0 MINGW64 /d/my_project (master)
```

2. 生成SSH公钥

- 生成命令: ssh-keygen -t rsa -C “邮箱” (注意: 这里需要敲3次回车)

```
MINGW64:/d/my_project
$ ssh-keygen -t rsa -C "heimaTest@itcast.cn"
Generating public/private rsa key pair.
Enter file in which to save the key (/c/Users/████████/ssh/id_rsa):
Created directory '/c/Users/████████/ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /c/Users/████████/ssh/id_rsa.
Your public key has been saved in /c/Users/████████/ssh/id_rsa.pub.
The key fingerprint is:
SHA256:UmrJVnGd7qq1toYFy6dx6C6w5mwnlXuWpXRiCTbGLEU heimaTest@itcast.cn
The key's randomart image is:
+---[RSA 3072]---+
| .E . . . .
| . o o |
| + o . |
| ..B= . .
| +B=S= . |
| .oo.X * . |
| + = &.. |
| .= + O.+.
| +oo =o+o. |
+---[SHA256]---+
```

- 查看命令: cat ~/.ssh/id_rsa.pub

```

.E . . .
. o o
+ o .
..B=.
+B=S=
.oo.X *
+= &..
.= + O.+
+oo =o+o.
+---[SHA256]-----
查看SSH公钥
admin@LAPTOP-LO4AUSVO MINGW64 /d/my_project (master)
$ cat ~/.ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAQABAAABgQDAAKciuj8gdxLUZcRhVkuLSBtknnFTYqF0lBMfWR8D
PRfCLtrJ2314nmRrPwCgOgDetsXxHsFBkUjuaRqCIdhjEEg7wTde917kv9zGviBcgecp/ErGFmw7Bts2
o27zf8ae2Q1BH3UfkP0sS0GGgX02atpKaKBEp7IitUzml678x9xptNfCrSSG+U1AS+5X208lcR5EZVD
yJRQWwvSRUoYBTo9FuWf7a3LngJbK/m9nEntKCVTnGKoni3mp8Zg4CdDzyNDqMnM2SNeI1HRb09+s53t
3q0z93GH8UYk5pN3DHUmQ78zGaBVmRuF9CrMr8xsUkTuBJw9Kf3WQkoV48bq3jskTuNByTBcgHN6b7CT
moKyNqIL1oSVDq1N9ZXN3Euj0Q2azzaPtmxhf8182tPTVAA4B1qM0k60h9MhpNMYkBAv06zJBmgsDr7
8pVWA8uNV/X2PgfdfOzbblDkR3spd3rQeX9KNMNscltZ/ofqkcqbmgmHyLGn7c+3x6nH0Fk= heimaTe
st@itcast.cn

admin@LAPTOP-LO4AUSVO MINGW64 /d/my_project (master)
$
```

3. 设置账户公钥



SSH公钥

使用SSH公钥可以让你在你的电脑和马云通讯的时候使用安全连接 (Git的remote要使用SSH地址)

你当前的SSH公钥数: 0

你还没有添加任何SSH公钥

添加公钥

标题: heimaTest@itcast.cn

公钥
把你的公钥粘贴到这里, 查看 怎样生成公钥

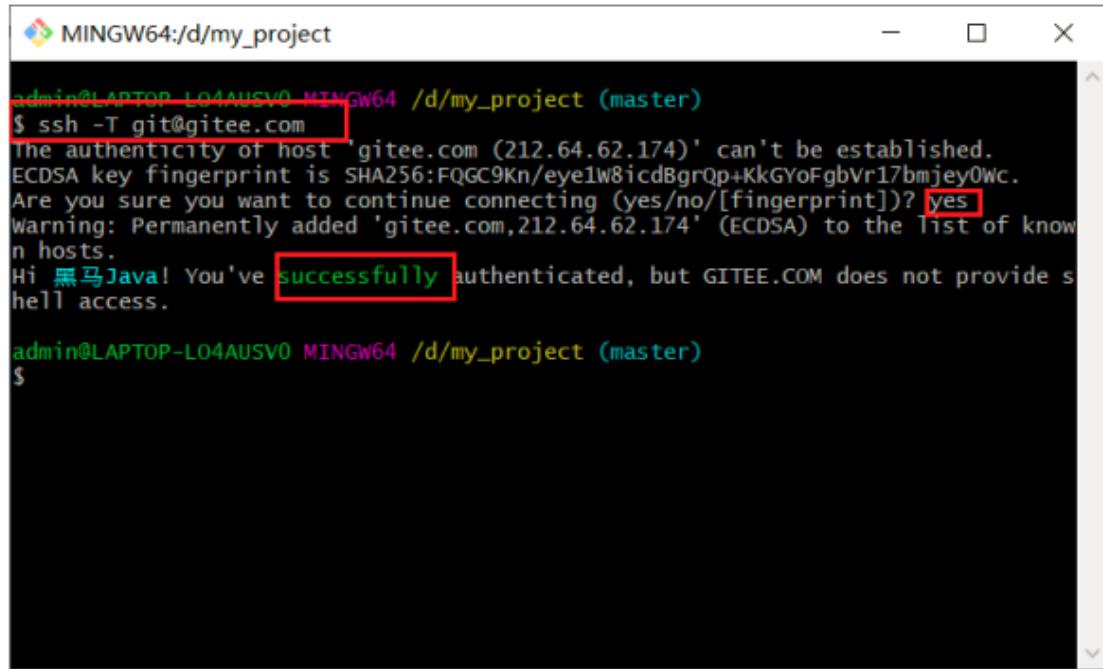
```

ssh-rsa AAAAB3NzaC1yc2EAAAQABAAABgQDAAKciuj8gdxLUZcRhVkuLSBtknnFTYqF0lBMfWR8DPRfCLtrJ2314nmRrPwCgOgDetsXxHsFBkUjuaRqCIdhjEEg7wTde917kv9zGviBcgecp/ErGFmw7Bts2
o27zf8ae2Q1BH3UfkP0sS0GGgX02atpKaKBEp7IitUzml678x9xptNfCrSSG+U1AS+5X208lcR5EZVD
yJRQWwvSRUoYBTo9FuWf7a3LngJbK/m9nEntKCVTnGKoni3mp8Zg4CdDzyNDqMnM2SNeI1HRb09+s53t
3q0z93GH8UYk5pN3DHUmQ78zGaBVmRuF9CrMr8xsUkTuBJw9Kf3WQkoV48bq3jskTuNByTBcgHN6b7CT
moKyNqIL1oSVDq1N9ZXN3Euj0Q2azzaPtmxhf8182tPTVAA4B1qM0k60h9MhpNMYkBAv06zJBmgsDr7
8pVWA8uNV/X2PgfdfOzbblDkR3spd3rQeX9KNMNscltZ/ofqkcqbmgmHyLGn7c+3x6nH0Fk= heimaTest@itcast.cn

```

4. 公钥测试

■ 命令: ssh -T git@gitee.com



```
admin@LAPTOP-L04AUSV0 MINGW64 /d/my_project (master)
$ ssh -T git@gitee.com
The authenticity of host 'gitee.com (212.64.62.174)' can't be established.
ECDSA key fingerprint is SHA256:FQGC9Kn/eyelw8icdBgrQp+KkGYoFgbVr17bmjey0Wc.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'gitee.com,212.64.62.174' (ECDSA) to the list of known hosts.
Hi 黑马Java! You've successfully authenticated, but GITEE.COM does not provide shell access.

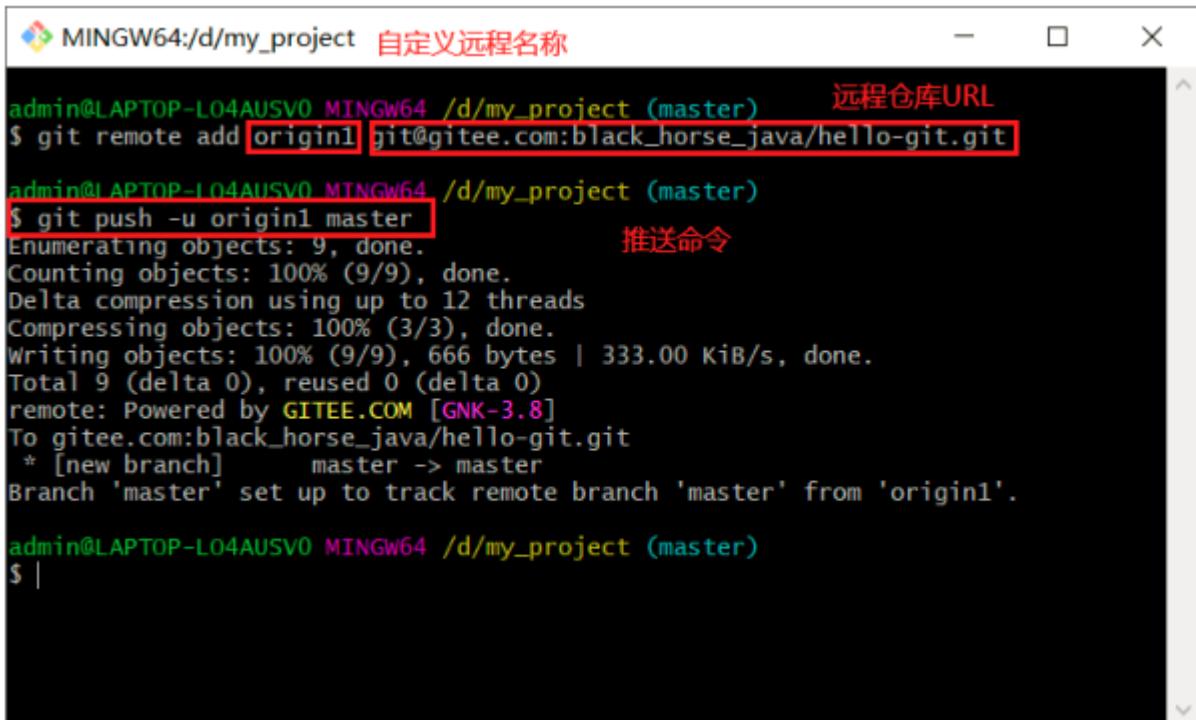
admin@LAPTOP-L04AUSV0 MINGW64 /d/my_project (master)
$
```

- 推送到远程仓库

- 步骤

1. 为远程仓库的URL (网址) , 自定义仓库名称
2. 推送

- 命令 git remote add 远程名称 远程仓库URL git push -u 仓库名称 分支名



```
admin@LAPTOP-L04AUSV0 MINGW64 /d/my_project (master) 远程仓库URL
$ git remote add origin1 git@gitee.com:black_horse_java/hello-git.git

admin@LAPTOP-L04AUSV0 MINGW64 /d/my_project (master)
$ git push -u origin1 master 推送命令
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 12 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (9/9), 666 bytes | 333.00 KiB/s, done.
Total 9 (delta 0), reused 0 (delta 0)
remote: Powered by GITEE.COM [GNK-3.8]
To gitee.com:black_horse_java/hello-git.git
 * [new branch] master -> master
Branch 'master' set up to track remote branch 'master' from 'origin1'.

admin@LAPTOP-L04AUSV0 MINGW64 /d/my_project (master)
$ |
```

你当前开源项目尚未选择许可证 (LICENSE)。点此选择并创建开源许可

暂无描述

3 次提交 3 个分支 0 个标签 0 个发行版 1 位贡献者

master + Pull Request + Issue 文件 Web IDE 挂件 克隆/下载

heimaTest 最后提交于 1 天前 commit third file test.txt

test.txt commit third file test.txt 1 天前

添加一个 README.md 文件，帮助感兴趣的人了解。添加 README

捐赠

5.5 先有远程仓库,本地为空(应用)

- 步骤

- 将远程仓库的代码, 克隆到本地仓库

克隆命令: git clone 仓库地址

- 创建新文件, 添加并提交到本地仓库

- 推送至远程仓库

- 项目拉取更新

拉取命令: git pull 远程仓库名 分支名

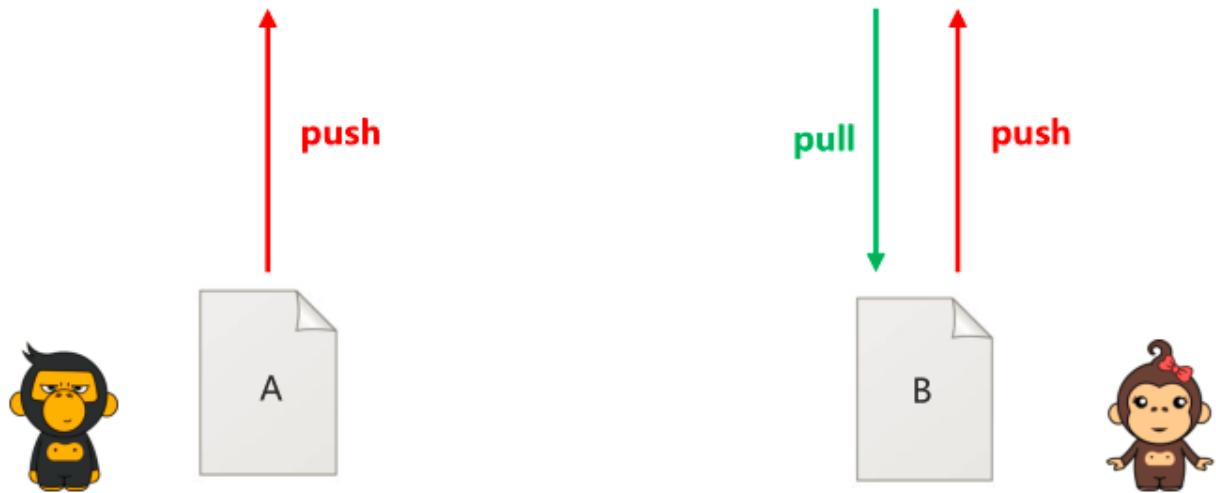
5.6 代码冲突(应用)

- 产生原因:

两个程序员操作同一个文件,其中一个程序员在修改文件后,push到远程仓库,另一个程序员应该先pull将最新的代码更新到本地仓库后,在修改代码,之后push到远程仓库,结果他没有先pull将最新的代码更新到本地仓库,而是直接将自己的代码push到远程仓库,这样就可能会导致代码冲突

远程仓库

代码托管平台（部署在公网上的一个网站）



```
admin@LAPTOP-L04AUSV0 MINGW64 /d/my_project (master)
$ git push -u origin1 master
To gitee.com:black_horse_java/test.git
 ! [rejected]      master -> master (fetch first)
error: failed to push some refs to 'git@gitee.com:black_horse_java/test.git'
hint: Updates were rejected because the remote contains work that you do
hint: not have locally. This is usually caused by another repository pushing
hint: to the same ref. You may want to first integrate the remote changes
hint: (e.g., 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
```

> 计算机 > Work (D:) > my_project

名称

- .git
- test.txt**
- test2.txt
- test3.txt

MINGW64:/d/my_project

```
admin@LAPTOP-L04AUSV0 MINGW64 /d/my_project (master)
$ git pull origin1 master
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0)
Unpacking objects: 100% (3/3), done.
From gitee.com:black_horse_java/test
 * branch            master      -> FETCH_HEAD
   490e3c9..83fa577  master      -> origin1/master
Auto-merging test.txt
CONFLICT (content): Merge conflict in test.txt
Automatic merge failed; fix conflicts and then commit the result.

admin@LAPTOP-L04AUSV0 MINGW64 /d/my_project (master|MERGING)
$ |
```

- 如何解决冲突

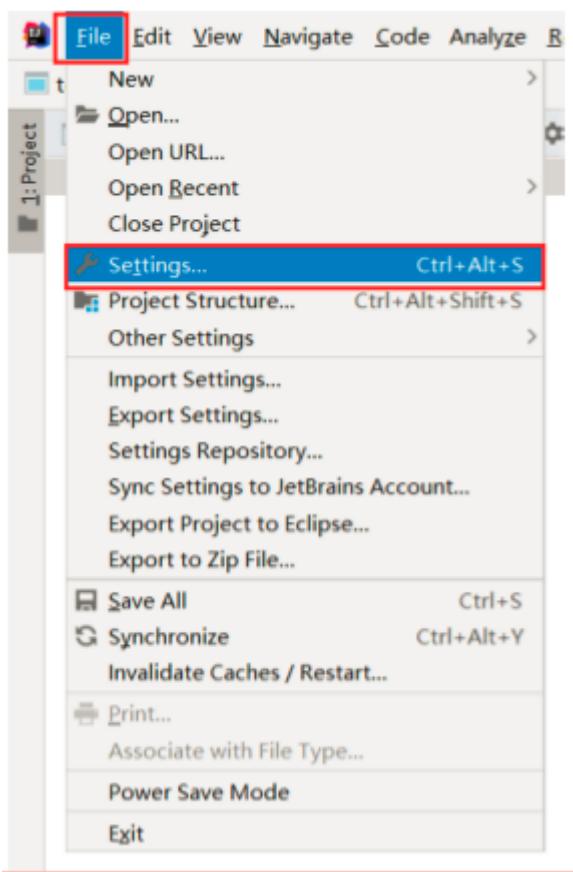
<<<<<和>>>>>中间的内容,就是冲突部分

1. 修改冲突行, 保存, 即可解决冲突。
2. 重新add冲突文件并commit到本地仓库, 重新push到远程

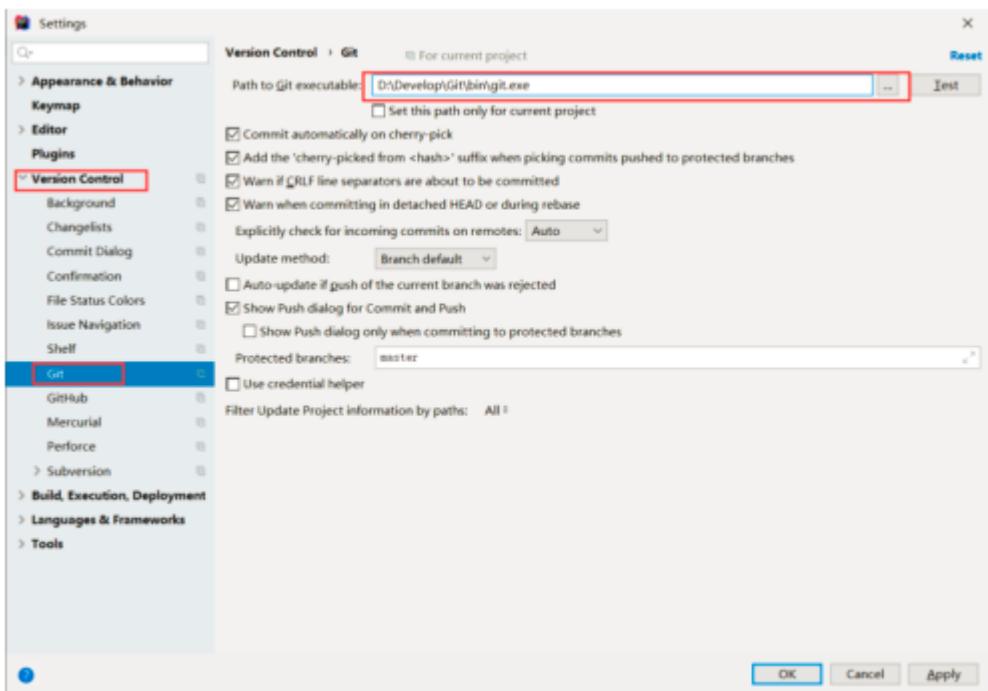
6. IDEA集成Git

6.1 IDEA中配置Git(应用)

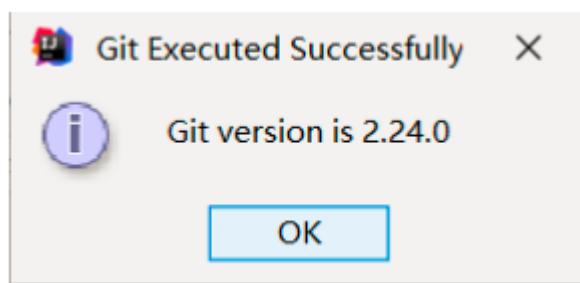
1. File -> Settings



2. Version Control -> Git -> 指定git.exe存放目录

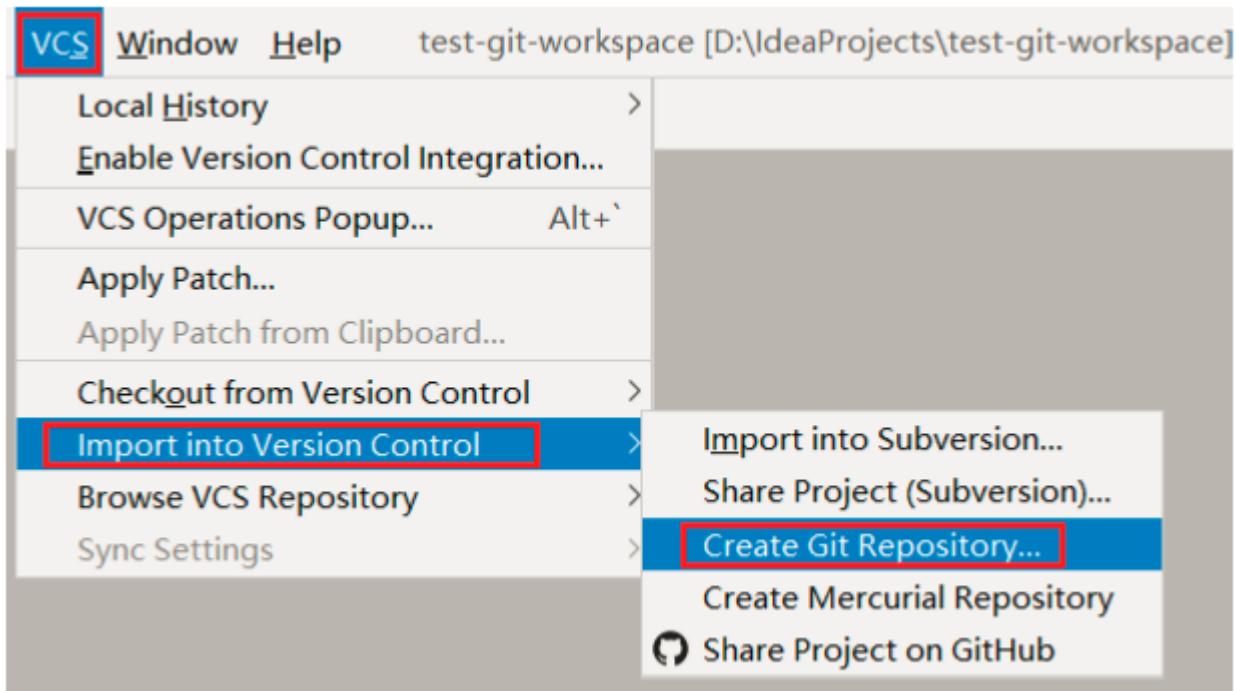


3. 点击Test测试



6.2 创建本地仓库(应用)

1. VCS->Import into Version Control->Create Git Repository



2. 选择工程所在的目录,这样就创建好本地仓库了



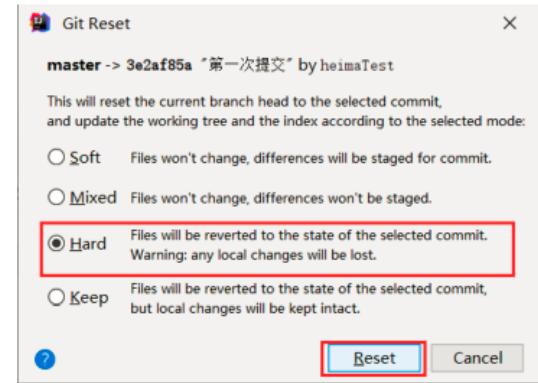
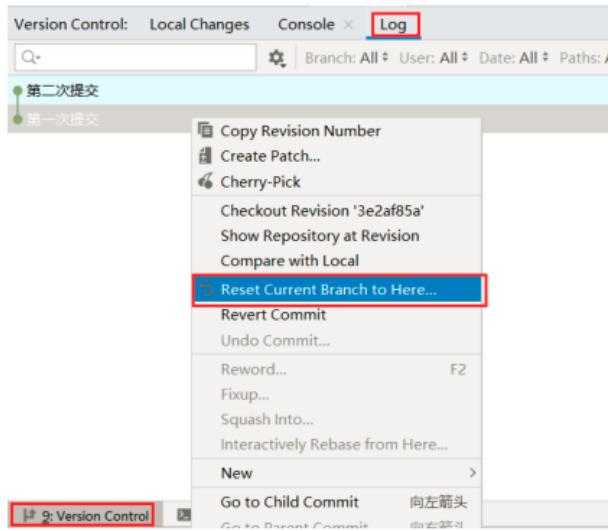
3. 点击git后边的对勾,将当前项目代码提交到本地仓库

注意: 项目中的配置文件不需要提交到本地仓库中,提交时,忽略掉即可



6.3 版本切换(应用)

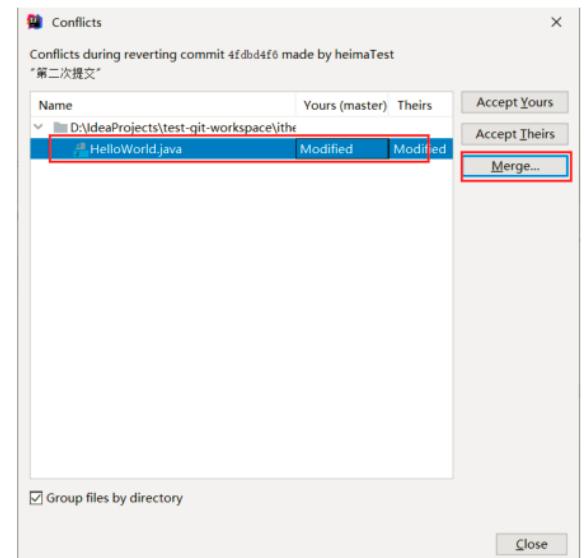
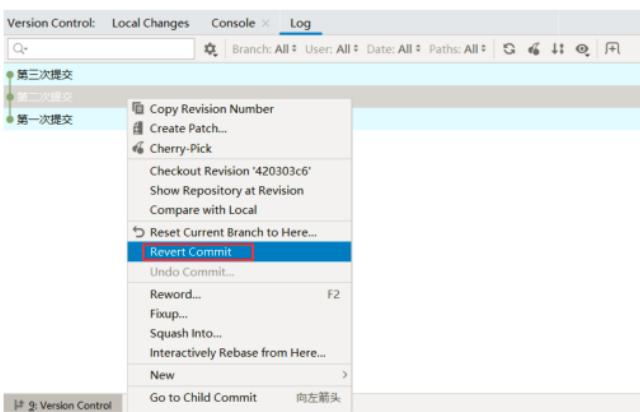
- 方式一: 控制台Version Control->Log->Reset Current Branch...->Reset
这种切换的特点是会抛弃原来的提交记录

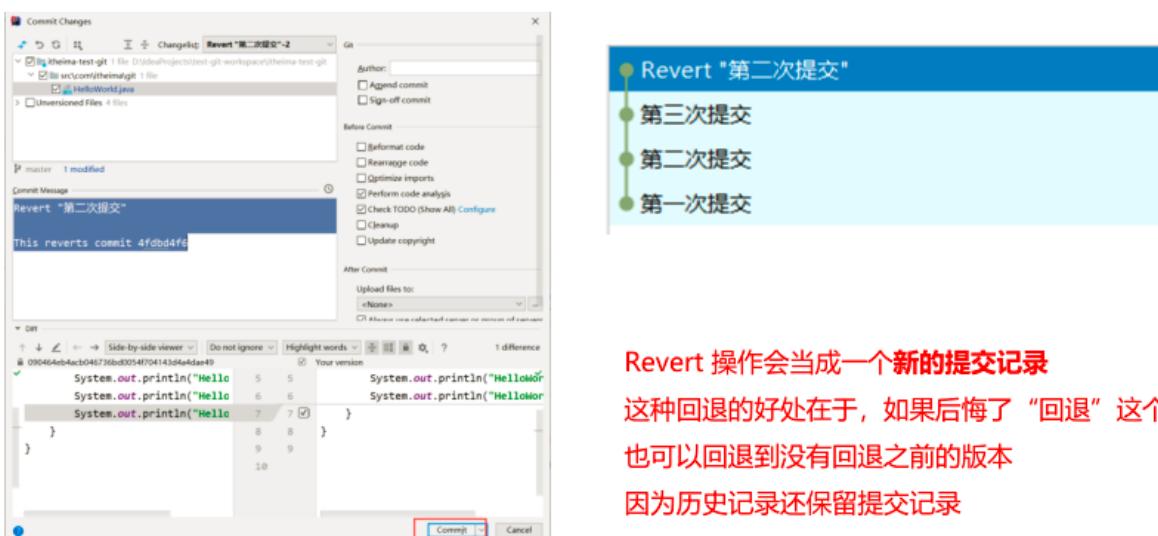
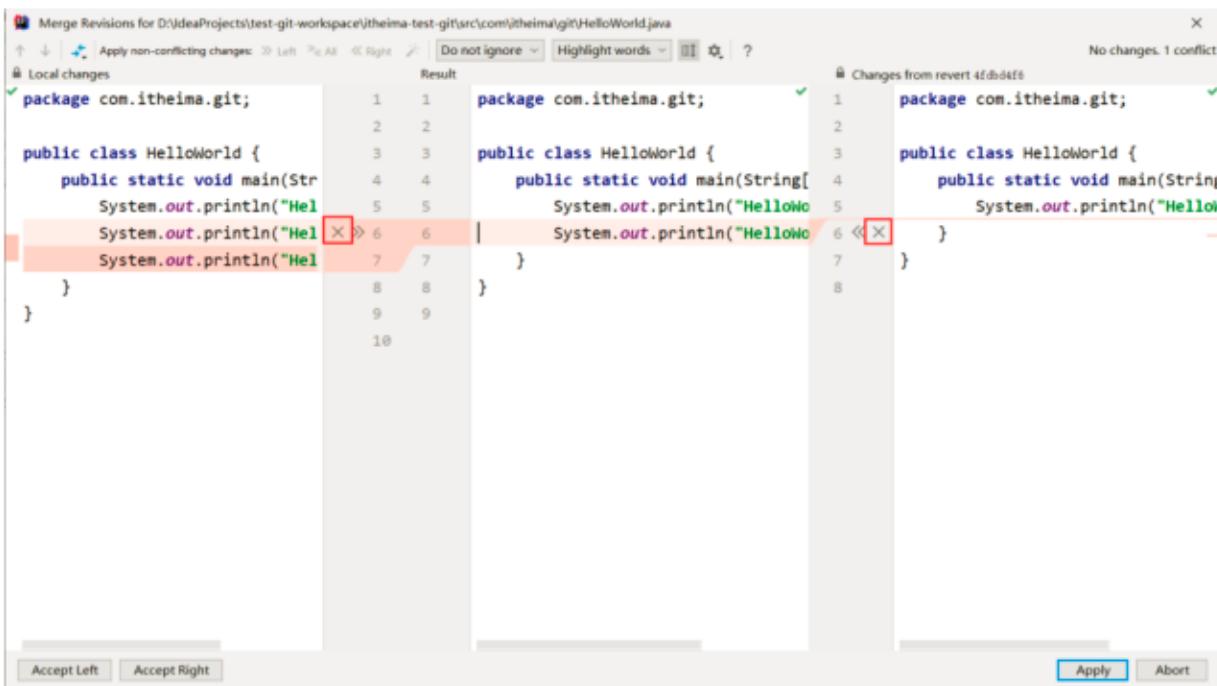


**Reset Head指针，会抛弃原来的提交记录
使Head指针强制指向指定的版本**

- 方式二:控制台Version Control->Log->Revert Commit->Merge->处理代码->commit

这种切换的特点是会当成一个新的提交记录,之前的提交记录也都保留





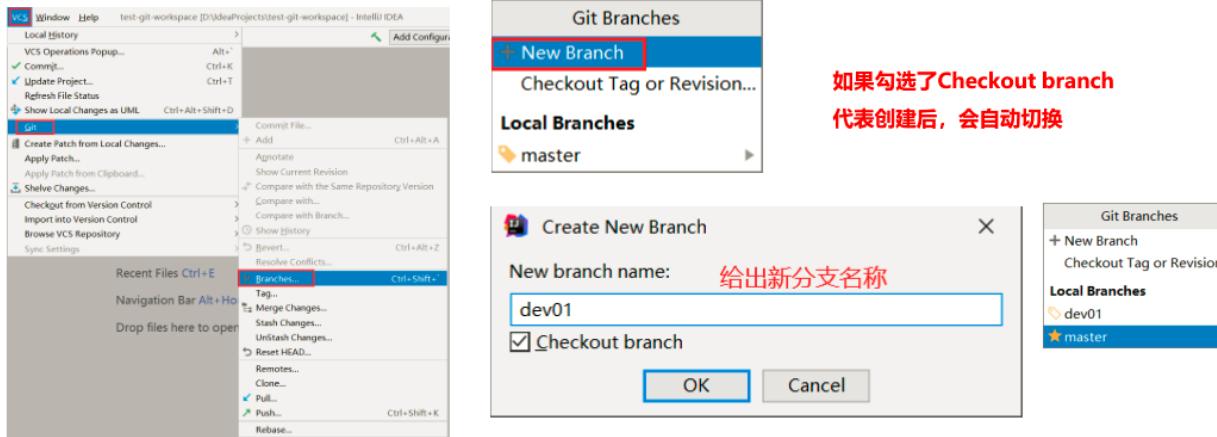
Revert 操作会当成一个新的提交记录

这种回退的好处在于，如果后悔了“回退”这个操作
也可以回退到没有回退之前的版本
因为历史记录还保留提交记录

6.4 分支管理(应用)

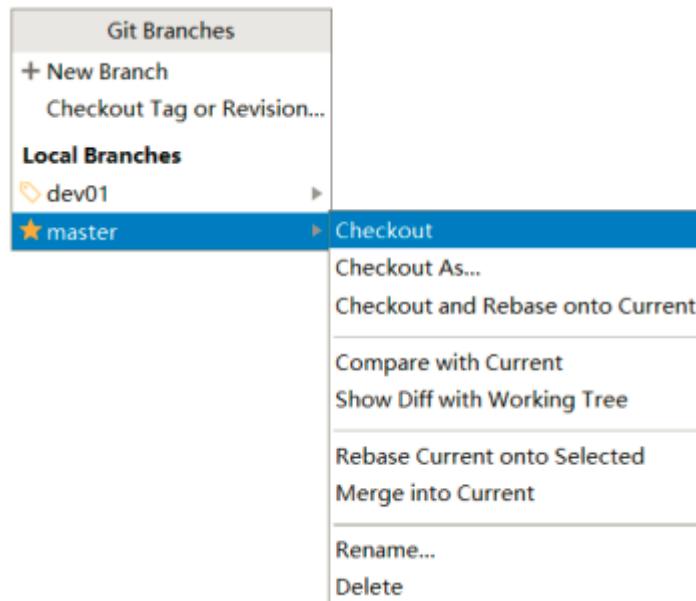
- 创建分支

VCS->Git->Branches->New Branch->给分支起名字->ok



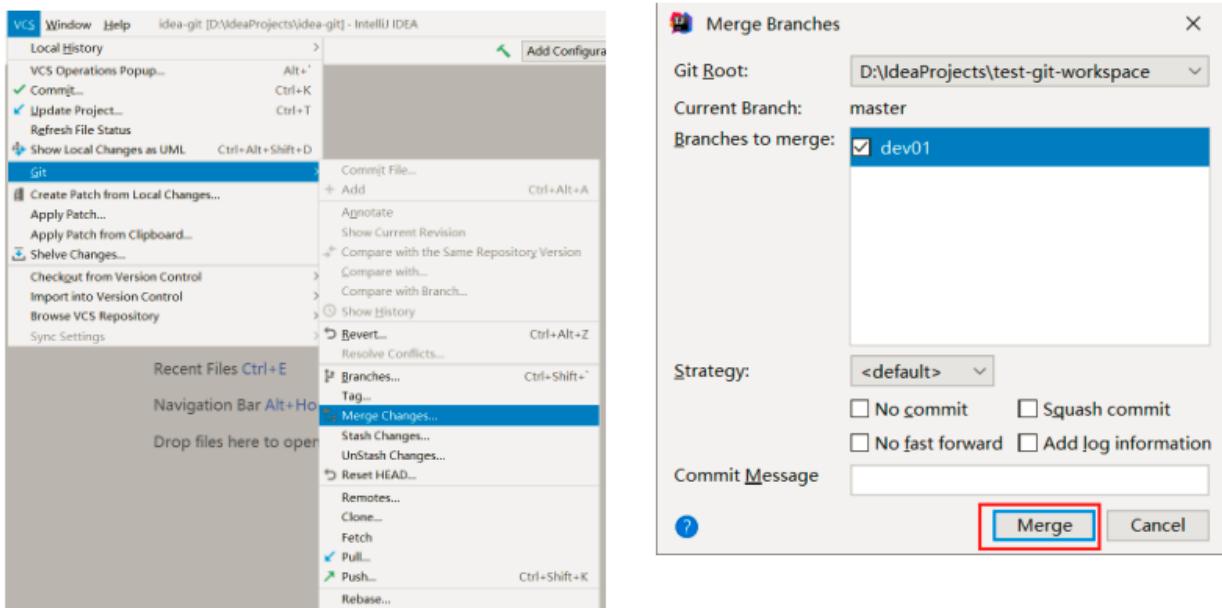
- 切换分支

idea右下角Git->选择要切换的分支->checkout



- 合并分支

VCS->Git->Merge changes->选择要合并的分支->merge

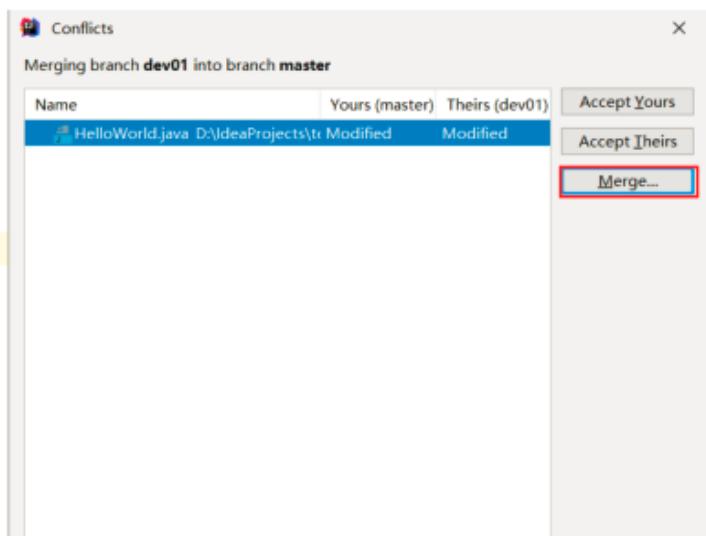


处理分支中的代码

```

public class HelloWorld {
    public static void main(String[] args) {
        <<<<< HEAD
        System.out.println("你好");
        =====
        System.out.println("HelloWorld");
        System.out.println("新增代码01");
        System.out.println("新增代码02");
        System.out.println("新增代码03");
        >>>>> dev01
    }
}

```



Merge Revisions for D:\IdeaProjects\test-git-workspace\itheima-test-git\src\com\itheima\git\HelloWorld.java

Your version, branch master

```

package com.itheima.git;

public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("你好");
    }
}

```

Changes from branch dev01 revision 8424f41d

```

package com.itheima.git;

public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("HelloWorld");
        System.out.println("新增代码01");
        System.out.println("新增代码02");
        System.out.println("新增代码03");
    }
}

```

Result

Accept Left Accept Right Apply Abort

Merge Revisions for D:\IdeaProjects\test-git-workspace\itheima-test-git\src\com\itheima\git\HelloWorld.java

Your version, branch master

```

package com.itheima.git;

public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("你好");
    }
}

```

All conflicts resolved

Changes from branch dev01, revision 8424f41d

```

package com.itheima.git;

public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("你好");
        System.out.println("HelloWorld");
        System.out.println("新增代码01");
        System.out.println("新增代码02");
        System.out.println("新增代码03");
    }
}

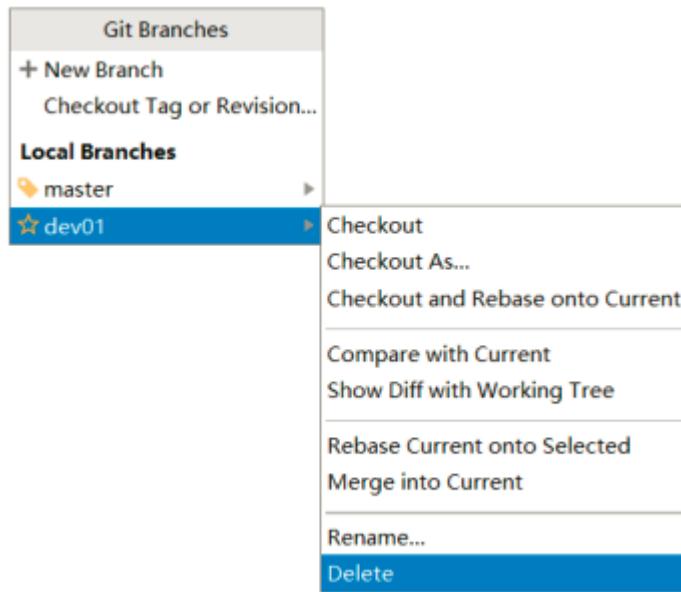
```

Result

Accept Left Accept Right Apply Abort

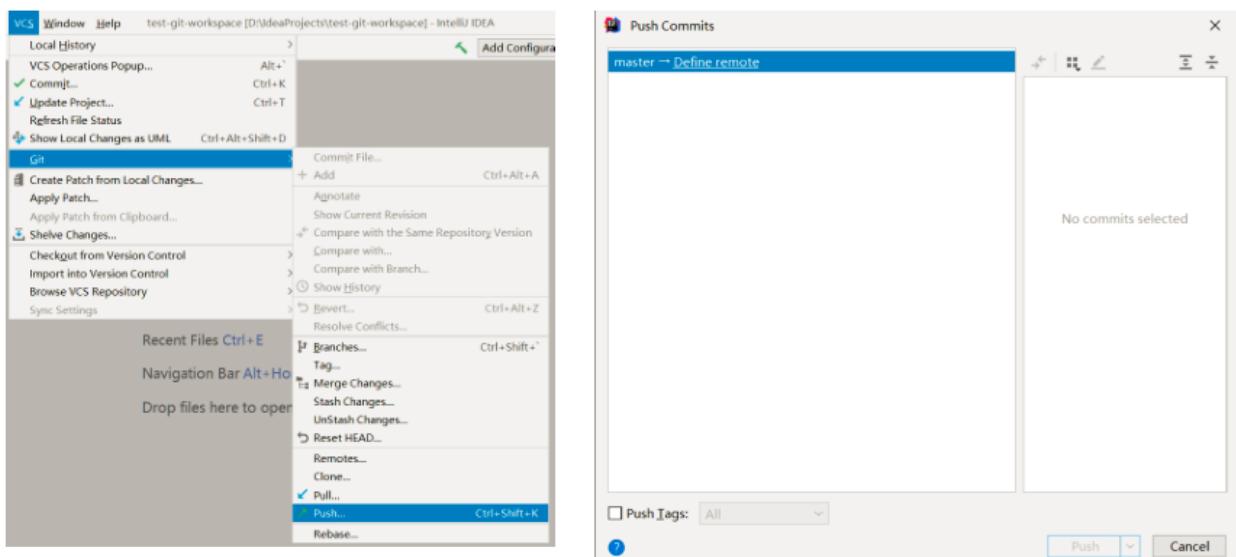
- **删除分支**

idea右下角->选中要删除的分支->Delete

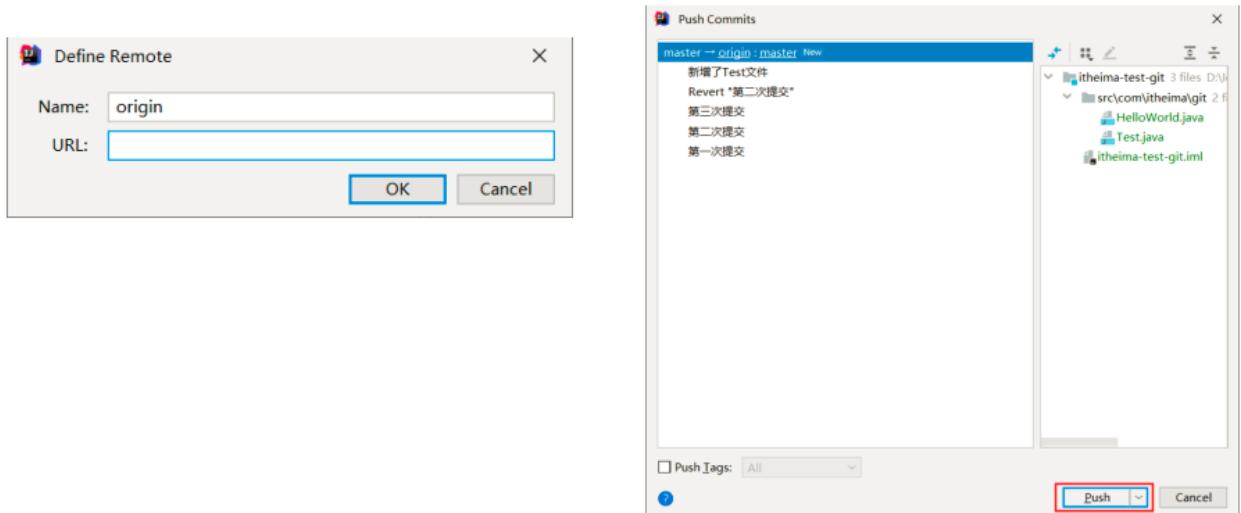


6.5本地仓库推送到远程仓库(应用)

1. VCS->Git->Push->点击master Define remote



2. 将远程仓库的路径复制过来->Push



6.6 远程仓库克隆到本地仓库(应用)

File->Close Project->Checkout from Version Control->Git->指定远程仓库的路径->指定本地存放的路径->clone

