



黑马程序员™
www.itheima.com

传智播客旗下
高端IT教育品牌



Git

目录 Contents

- ◆ Git介绍
- ◆ Git下载和安装
- ◆ Git操作入门
- ◆ Git版本管理
- ◆ 远程仓库
- ◆ IDEA集成Git

版本控制

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)

你之前写的有一版还行
就用那个了



开发中存在的麻烦

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



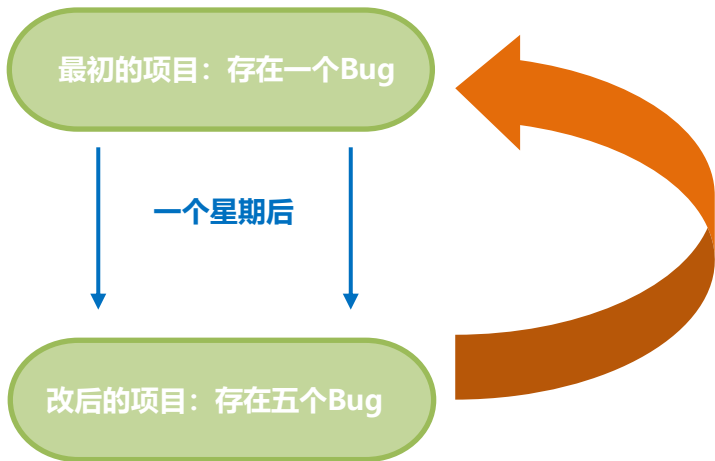
当初若是备份了代码
老夫也不会落得如此田地
痛哉！哀哉！

● 代码备份很重要

代码备份不仅要备份到本地，还需要备份到云端

开发中存在的麻烦

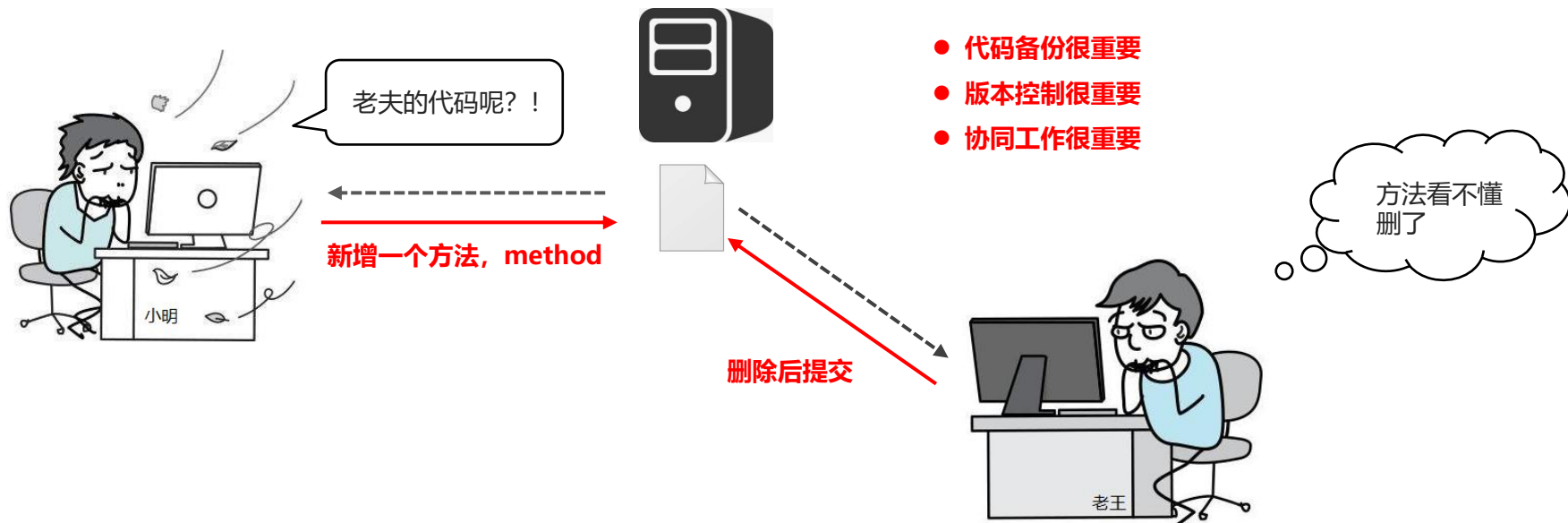
老王需要在项目中加入一个很复杂的功能，一边尝试，一边修改代码，就这样摸索了一个星期。
可是这被改得面目全非的代码已经回不到从前了。



- 代码备份很重要
- 版本控制很重要

开发中存在的麻烦

小明和老王先后从文件服务器上下载了同一个文件



开发中存在的麻烦

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

- 代码备份很重要
- 版本控制很重要
- 协同工作很重要
- 责任追溯很重要



开发中存在的麻烦

- 代码备份很重要
- 版本控制很重要
- 协同工作很重要
- 责任追溯很重要



为了避免开发中存在的这些问题
我们将使用专业的代码版本控制系统



git

Git与SVN对比

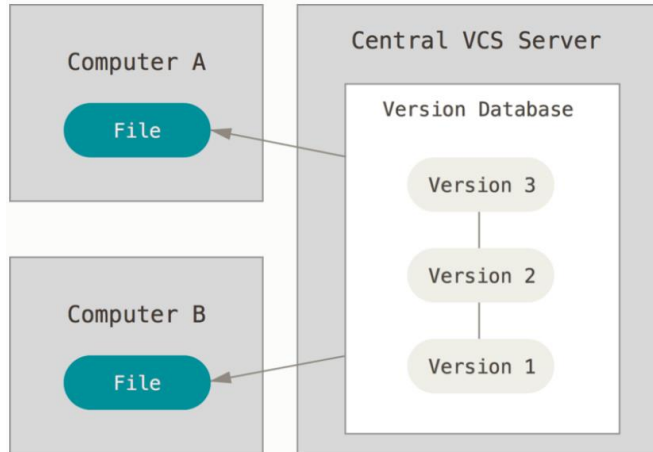
SVN是集中式版本控制系统，版本库是集中放在中央服务器的，而开发人员工作的时候，用的都是自己的电脑，所以首先要从中央服务器下载最新的版本，然后开发，开发完后，需要把自己开发的代码提交到中央服务器。

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

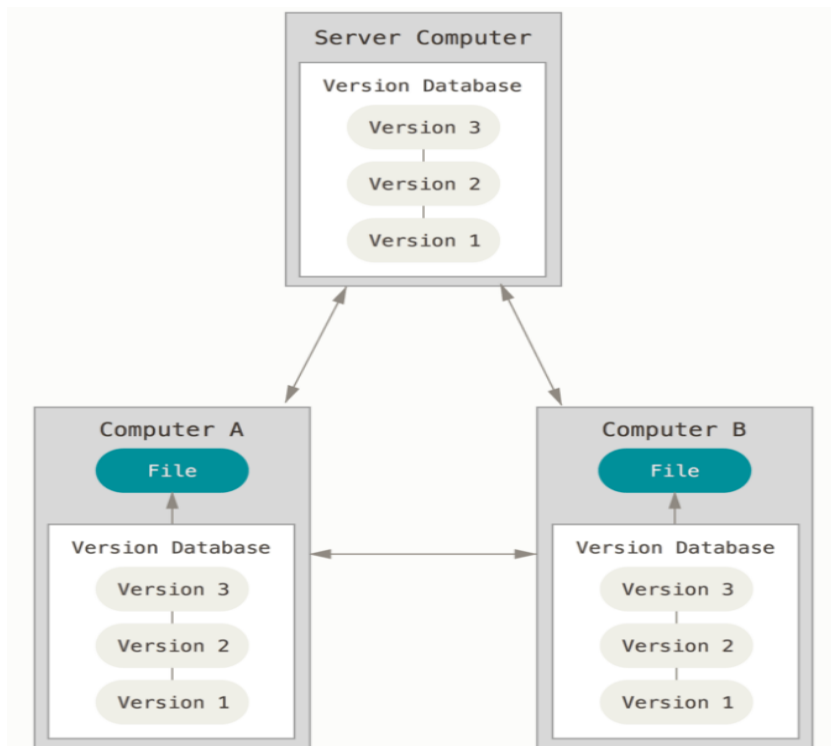
这意味着，你可能失去了该项目的历史记录，这是毁灭性的。



Git与SVN对比

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。



开发中存在的麻烦

- 代码备份很重要
- 版本控制很重要
- 协同工作很重要
- 责任追溯很重要

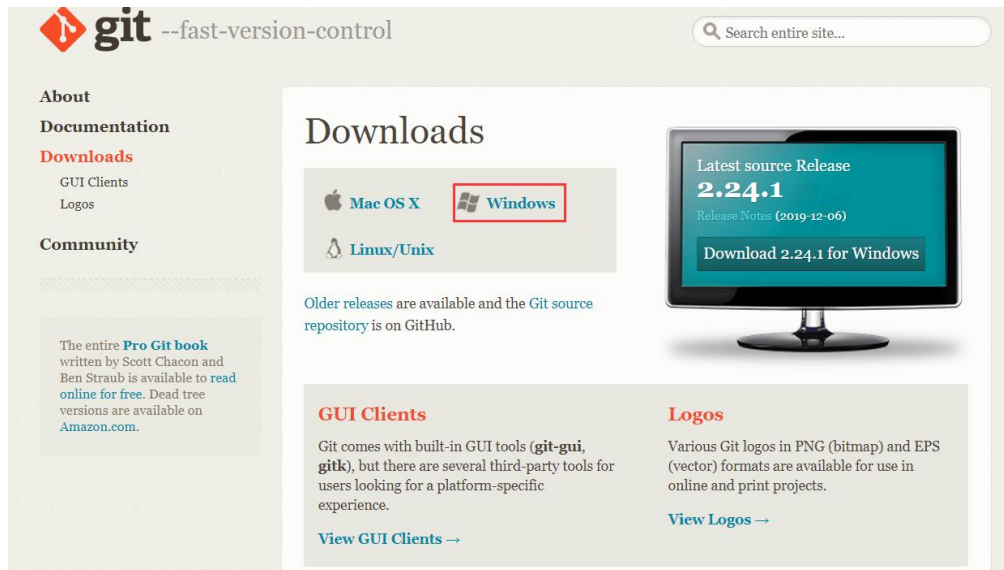


目录 Contents

- ◆ Git介绍
- ◆ Git下载和安装
- ◆ Git操作入门
- ◆ Git版本管理
- ◆ 远程仓库
- ◆ IDEA集成Git

Git下载

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



The screenshot shows the Git website's Downloads page. The header features the Git logo and the tagline "--fast-version-control", along with a search bar. The left sidebar contains navigation links: About, Documentation, Downloads (highlighted), GUI Clients, Logos, and Community. Below these is a promotional text for the Pro Git book. The main content area is titled "Downloads" and includes a section for "Latest source Release 2.24.1" with a button to "Download 2.24.1 for Windows". Below this, there are sections for "GUI Clients" and "Logos".

git --fast-version-control

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

Latest source Release
2.24.1
Release Notes (2019-12-06)
[Download 2.24.1 for Windows](#)

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 →](#)

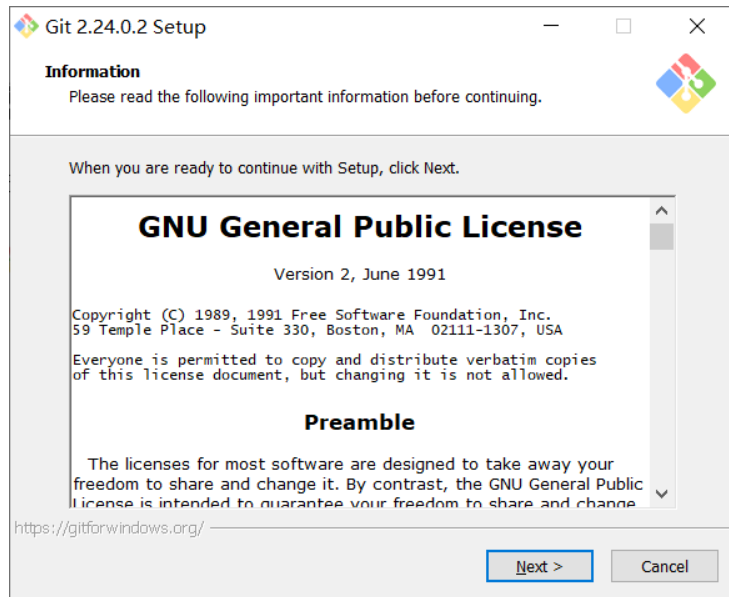
Logos

Various Git logos in PNG (bitmap) and EPS (vector) formats are available for use in online and print projects.
[View Logos →](#)

Git安装

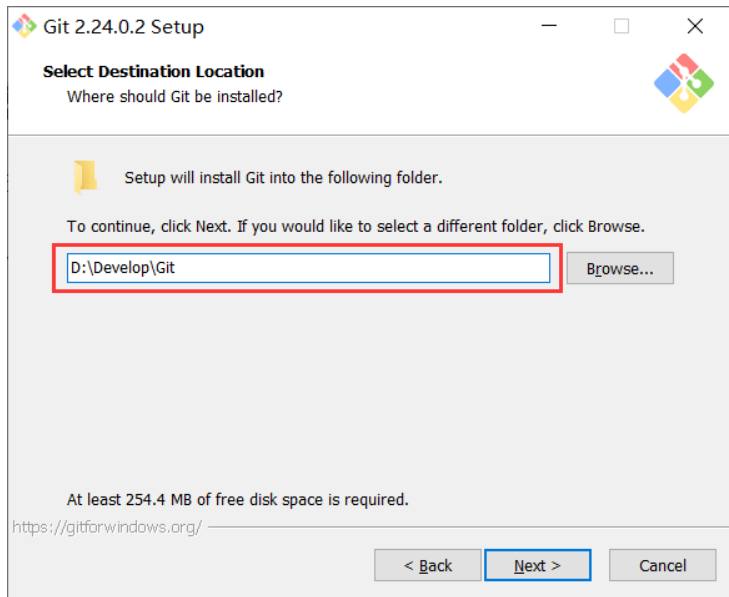
1. 双击安装包，进入安装向导界面

Git-2.24.0.2-64-bit.exe



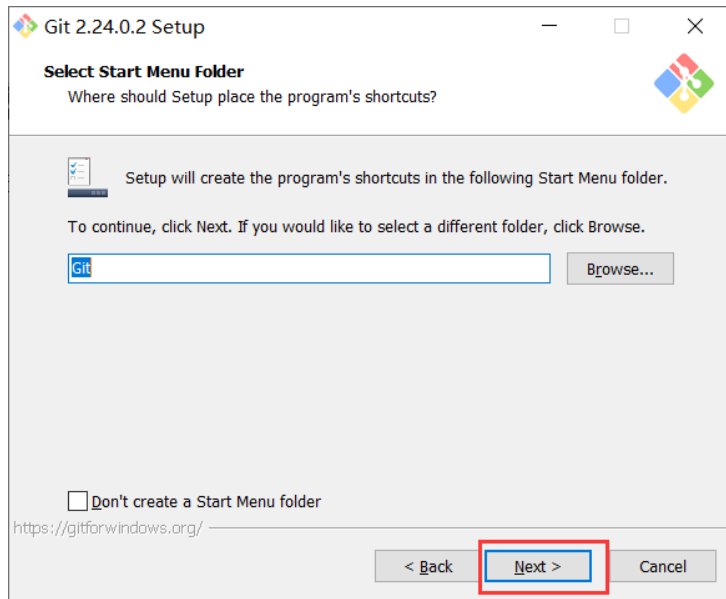
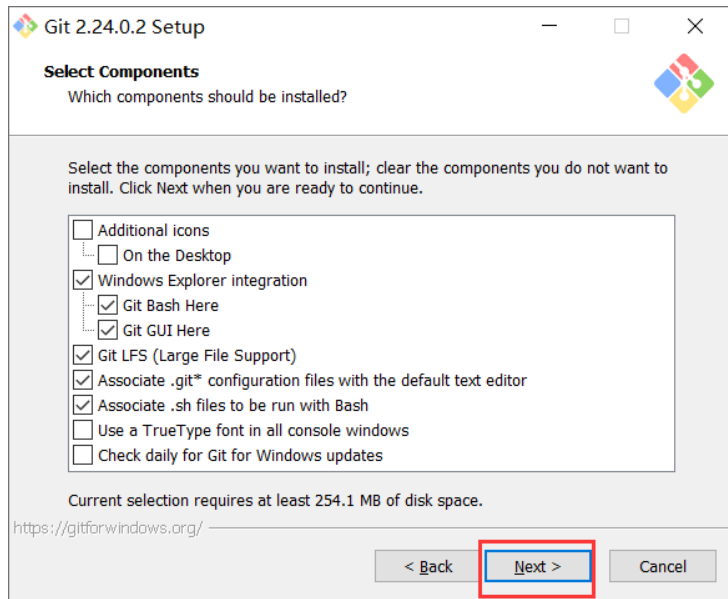
Git安装

2. 指定安装目录



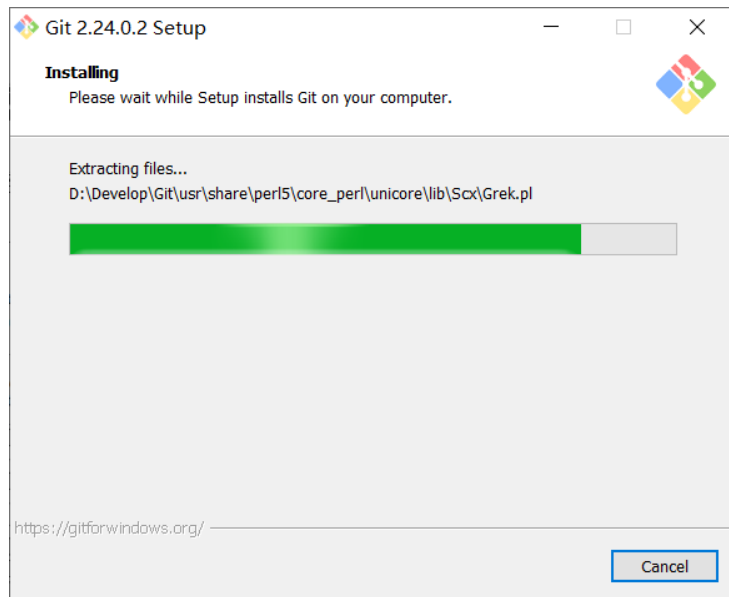
Git安装

3. 一路next下一步



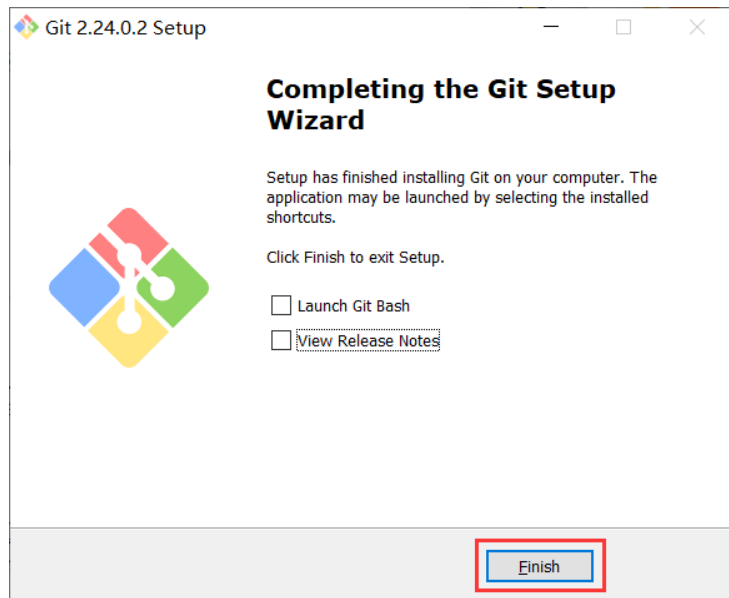
Git安装

4. 等待安装



Git安装

5. 安装完成



Git安装

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。

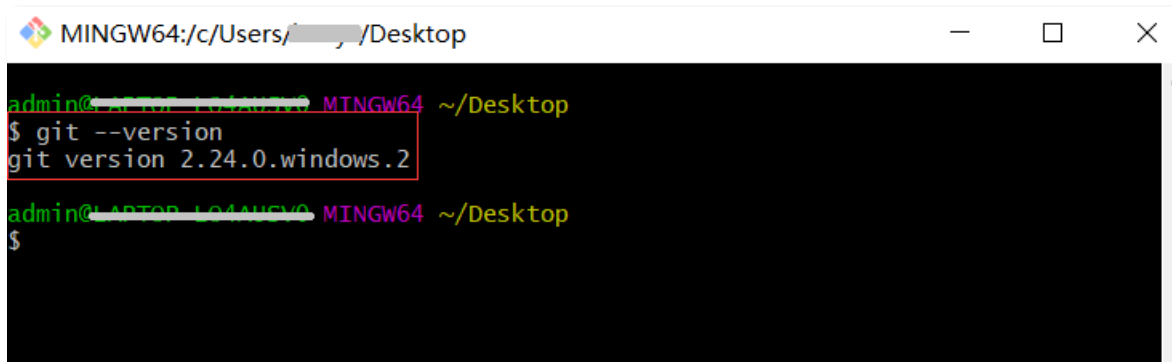


Git GUI: Git提供的图形界面工具

Git Bash: Git提供的命令行工具

Git安装

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



```
MINGW64:/c/Users/[redacted]/Desktop
admin@ARTOP-LO44UCV0 MINGW64 ~/Desktop
$ git --version
git version 2.24.0.windows.2
admin@ARTOP-LO44UCV0 MINGW64 ~/Desktop
$
```

TortoiseGit安装

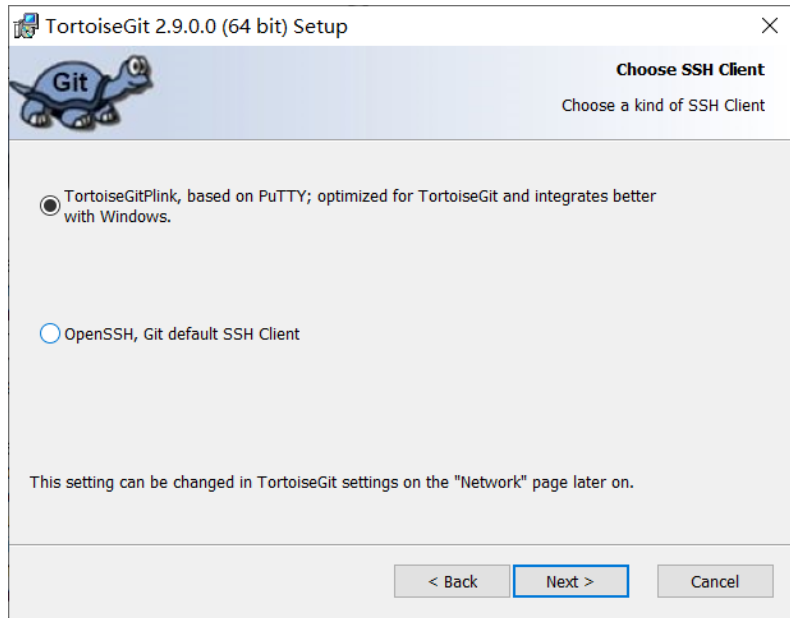
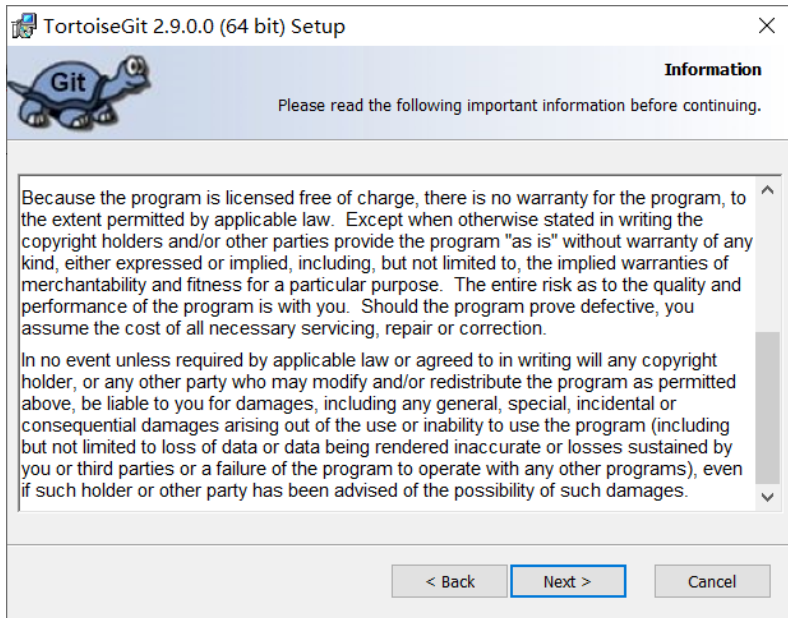
1. 双击安装包，进入安装向导界面

TortoiseGit-2.9.0.0-64bit.msi



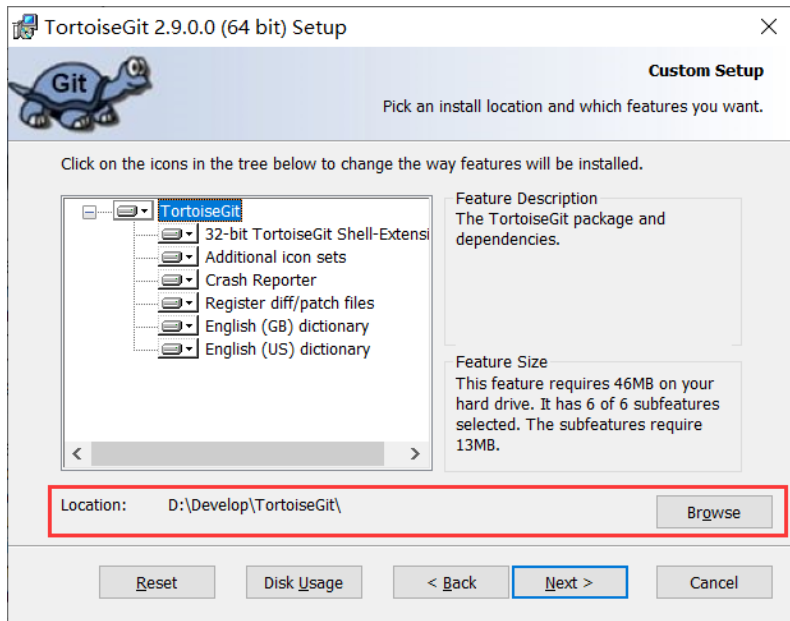
TortoiseGit安装

2. 一路next下一步



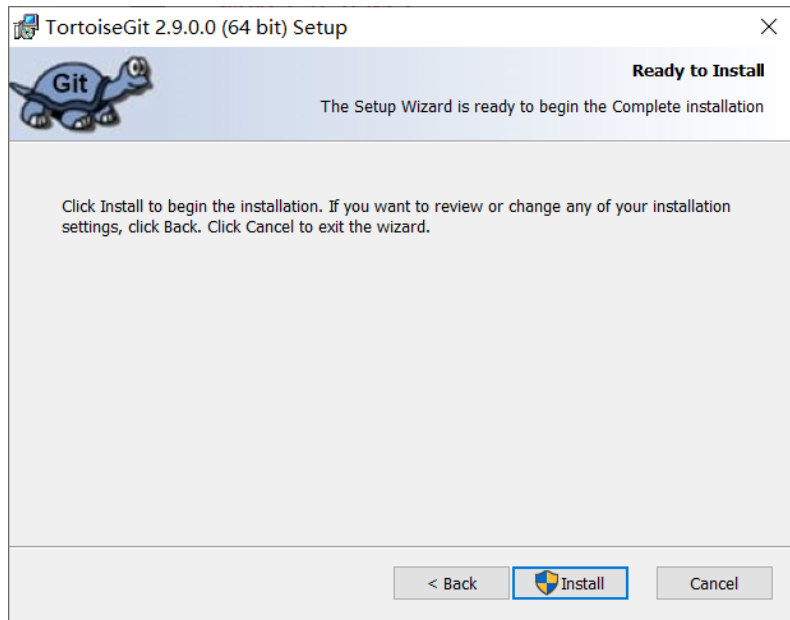
TortoiseGit安装

3. 指定安装目录



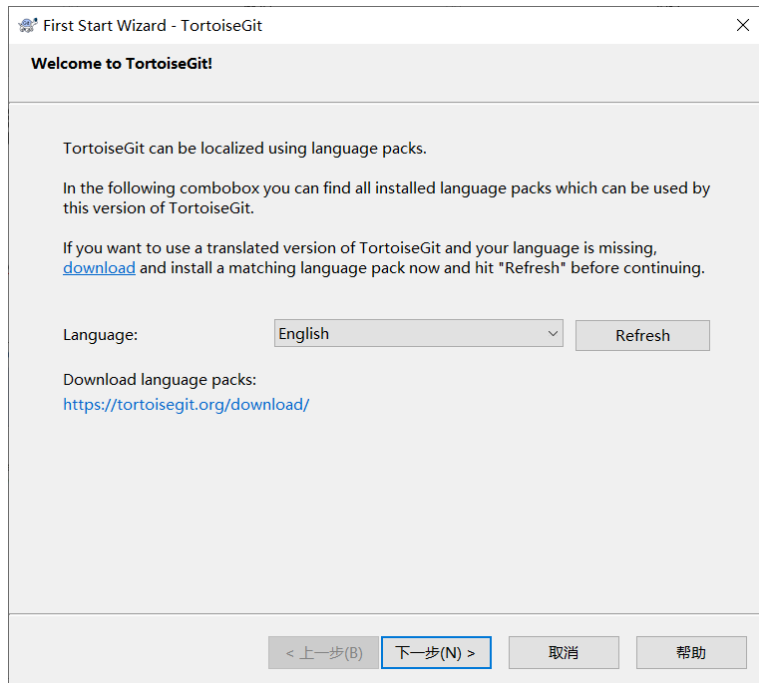
TortoiseGit安装

4. 安装



TortoiseGit安装

5. 配置



TortoiseGit安装

5. 配置

First Start Wizard - TortoiseGit

Configure user information

Git requires that you set up a user name and email address. Both are used as meta data for your commits (not for authentication).

Name: 开发者名称

Email: 开发者邮箱

These settings will be stored to your global git configuration (%HOME%/.gitconfig) and will be used for all your git repositories as a default.

☐ Don't store these settings now.

< 上一步(B) 下一步(N) > 取消 帮助

TortoiseGit安装

5. 配置

First Start Wizard - TortoiseGit

Configure git.exe

TortoiseGit requires a git.exe for its operations. TortoiseGit tries to automatically detect a working git.exe, but if that doesn't work or you want to use a different one please specify the path manually!

Git.exe Path: Git软件的安装目录 ...

Extra PATH:

Recommended: Git for Windows
<https://git-for-windows.github.io/>

< 上一步(B) 下一步(N) >

TortoiseGit安装

5. 配置

First Start Wizard - TortoiseGit

Authentication and credential store

SSH (URLs look like "git@example.com")

TortoiseGitPlink is the recommended as SSH client. If you don't have a key pair yet, you should generate one. Keep the private one in a safe place and set up the public key on your hosting platform. Use the PuTTY authentication agent for caching the password (done automatically if a PuTTY key is configured for a remote). For advanced tips & tricks see our [manual](#) and [FAQ](#).

TortoiseGitPlink Generate PuTTY key pair

HTTP (URLs start with "http://" or "https://")

By default Git does not save/cache credentials. However, you can configure a credential helper (recommended) or manually use %HOME%/_netrc.

Credential helper: manager

☐ Don't store these settings now. Advanced...

These settings will be stored to your global git configuration (%HOME%/.gitconfig) and will be used for all your git repositories as a default.

< 上一步(B) 完成 取消 帮助

TortoiseGit安装

5. 配置

First Start Wizard - TortoiseGit

Authentication and credential store

SSH (URLs look like "git@example.com")

TortoiseGitPlink is the recommended as SSH client. If you don't have a key pair yet, you should generate one. Keep the private one in a safe place and set up the public key on your hosting platform. Use the PuTTY authentication agent for caching the password (done automatically if a PuTTY key is configured for a remote). For advanced tips & tricks see our [manual](#) and [FAQ](#).

TortoiseGitPlink Generate PuTTY key pair

HTTP (URLs start with "http://" or "https://")

By default Git does not save/cache credentials. However, you can configure a credential helper (recommended) or manually use %HOME%/_netrc.

Credential helper: manager

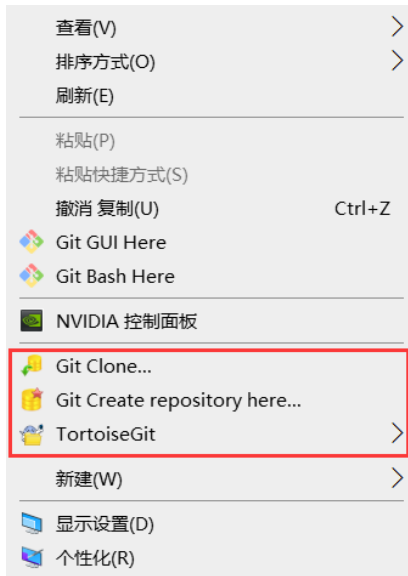
☐ Don't store these settings now. Advanced...

These settings will be stored to your global git configuration (%HOME%/.gitconfig) and will be used for all your git repositories as a default.

< 上一步(B) 完成 取消 帮助


TortoiseGit安装

5. 配置



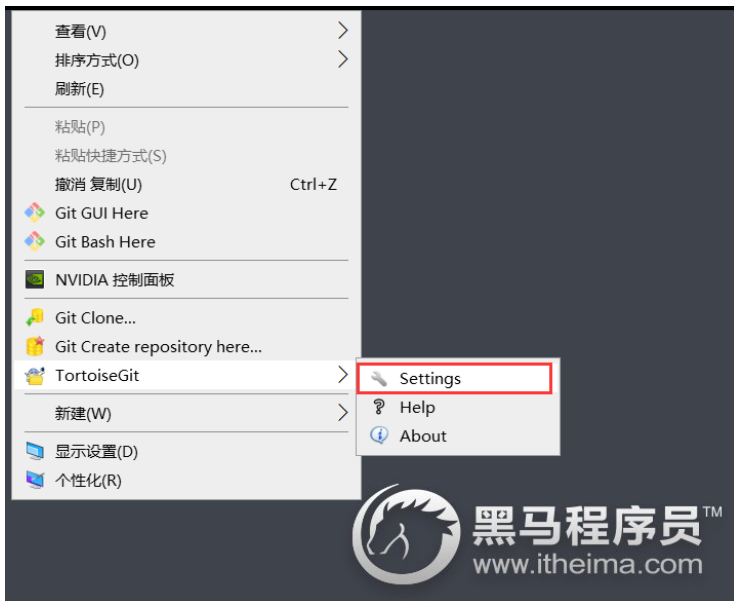
TortoiseGit安装

6. 安装中文语言包

 TortoiseGit-LanguagePack-2.9.0.0-64bit-zh_CN.msi

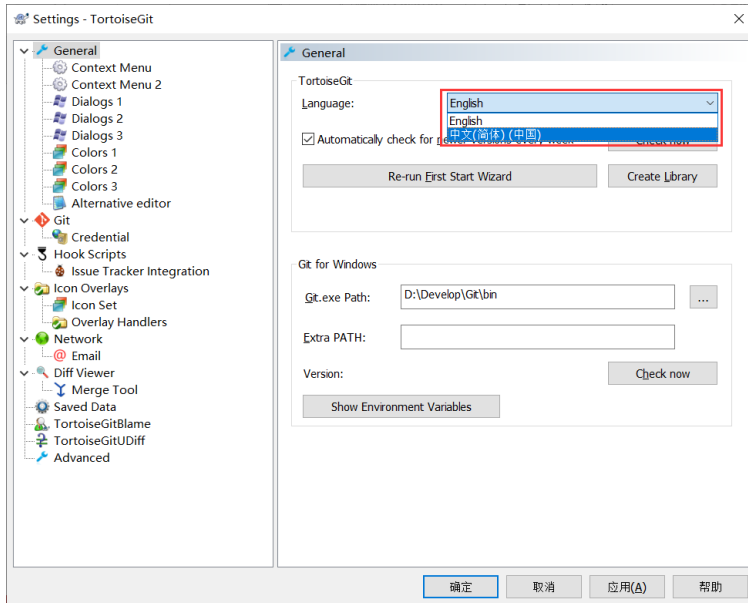
TortoiseGit安装

6. 安装中文语言包



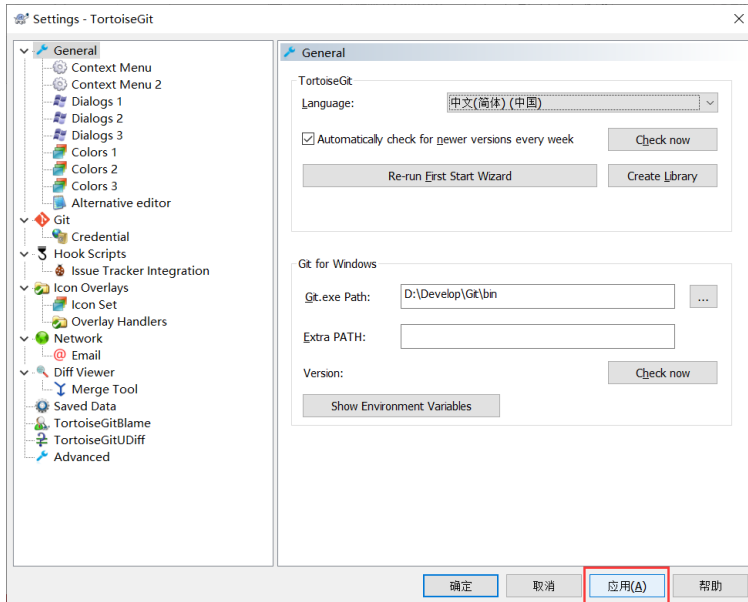
TortoiseGit安装

6. 安装中文语言包



TortoiseGit安装

6. 安装中文语言包



TortoiseGit安装

6. 安装中文语言包



目录 Contents

- ◆ Git介绍
- ◆ Git下载和安装
- ◆ Git操作入门
- ◆ Git版本管理
- ◆ 远程仓库
- ◆ IDEA集成Git

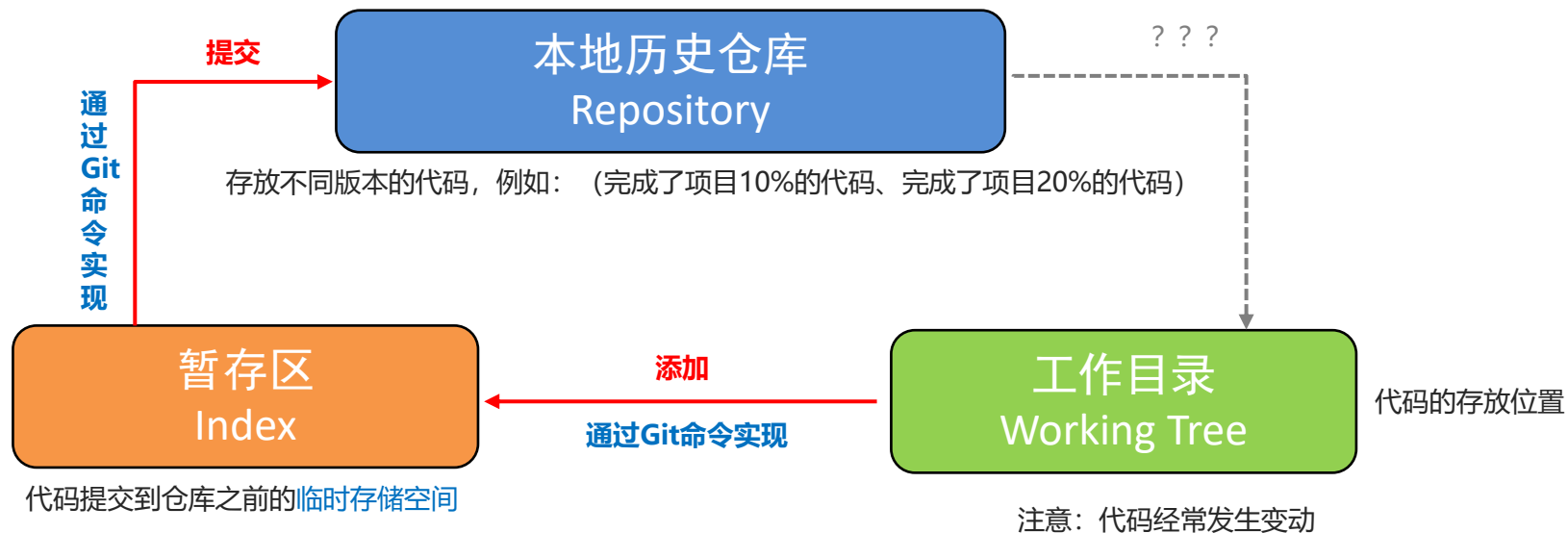
Git基本工作流程

- 本地仓库
- 远程仓库

目标：使用Git工具来管理代码

Git基本工作流程

- 本地仓库



Git基本工作流程

- Git常用命令

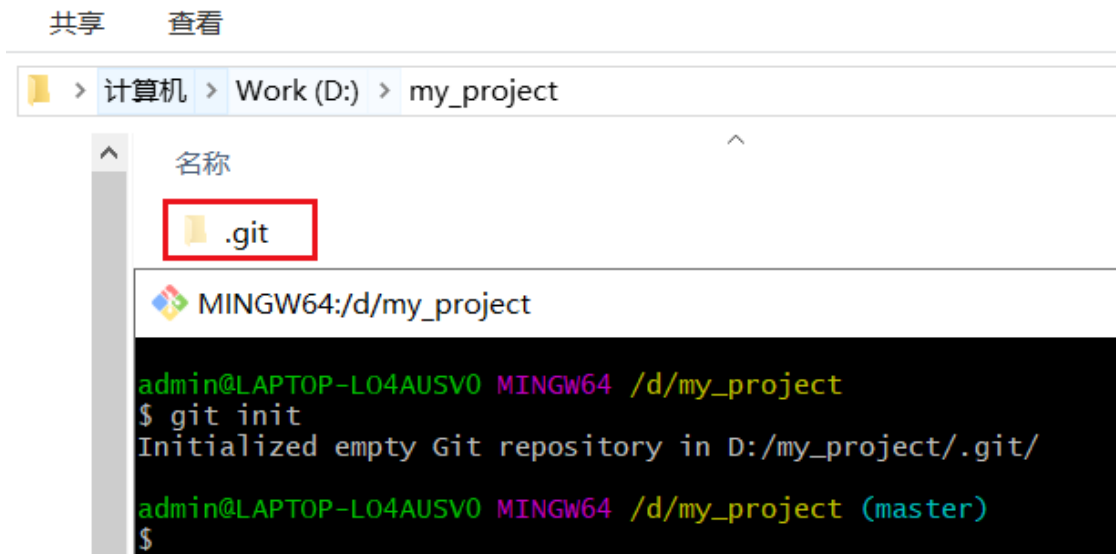
命令	作用
git init	初始化, 创建 git 仓库
git status	查看 git 状态 (文件是否进行了添加、提交操作)
git add	添加, 将指定文件添加到暂存区
git commit	提交, 将暂存区文件提交到历史仓库
git log	查看日志 (git 提交的历史日志)

Git基本工作流程

- 步骤：
 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. 使用 status 命令，查看状态
 4. 使用 add 命令添加，并查看状态
 5. 使用 commit 命令，提交到本地历史仓库
 6. 使用 log 命令，查看日志
 7. 修改 test.txt 文件
 8. 添加并提交，查看日志

Git基本工作流程

1. 创建工作目录、初始化本地 git 仓库



Git基本工作流程

2. 新建一个 test.txt 文件 (暂不执行添加操作)
3. 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)
$ |
```

Git基本工作流程

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

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)
$
```

Git基本工作流程

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

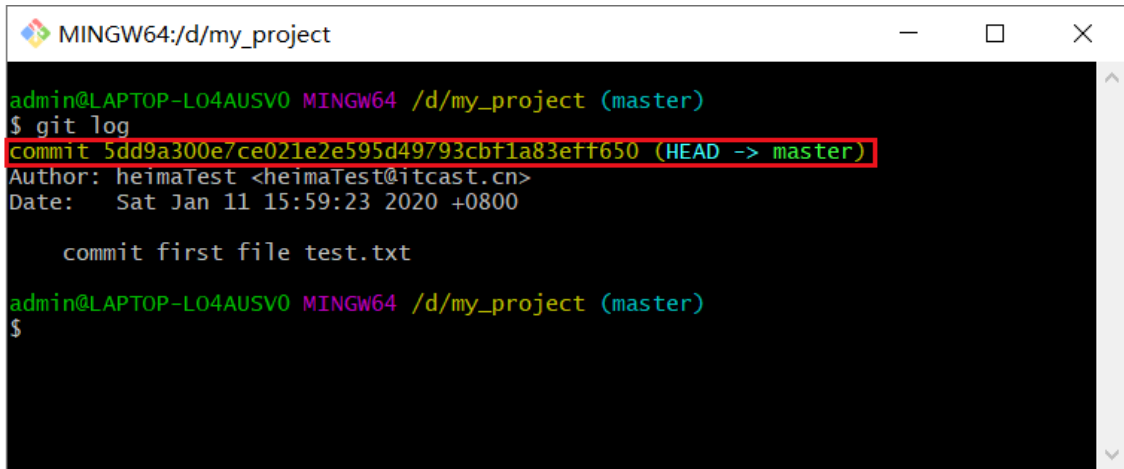
admin@LAPTOP-L04AUSV0 MINGW64 /d/my_project (master)
$ git commit -m 'commit first file test.txt'
[master (root-commit) 5dd9a30] commit first file test.txt
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 test.txt

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

git commit -m '提交时, 携带的描述信息'

Git基本工作流程

6. 使用 log 命令，查看日志

A terminal window titled 'MINGW64:/d/my_project' with standard window controls. The terminal shows a user running 'git log' in a directory. The output displays a single commit with a red box highlighting the commit hash and branch information. The commit message is also visible.

```
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)
$
```

Git基本工作流程

7. 修改 test.txt 文件内容

> 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

Git基本工作流程

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-LO4AUSV0 MINGW64 /d/my_project (master)
$ git add test.txt

admin@LAPTOP-LO4AUSV0 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-LO4AUSV0 MINGW64 /d/my_project (master)
$ git log
commit 3471d4760fc40be03b6172b975385c25144e120b (HEAD -> master)
Author: heima test <heimatest@itcast.cn>
Date: Sat Jan 11 16:15:04 2020 +0800

    commit second file test.txt

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

    commit first file test.txt

admin@LAPTOP-LO4AUSV0 MINGW64 /d/my_project (master)
$
```

注意: 白色框中的内容

Git基本工作流程

图形化工具

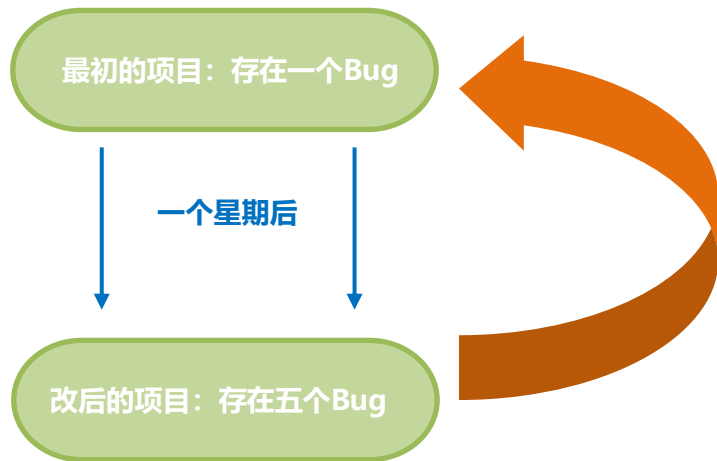
- 步骤:

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. 使用 status 命令，查看状态
4. 使用 add 命令添加，并查看状态
5. 使用 commit 命令，提交到本地历史仓库
6. 使用 log 命令，查看日志
7. 修改 test.txt 文件
8. 添加并提交，查看日志

目录 Contents

- ◆ Git介绍
- ◆ Git下载和安装
- ◆ Git操作入门
- ◆ Git版本管理
- ◆ 远程仓库
- ◆ IDEA集成Git

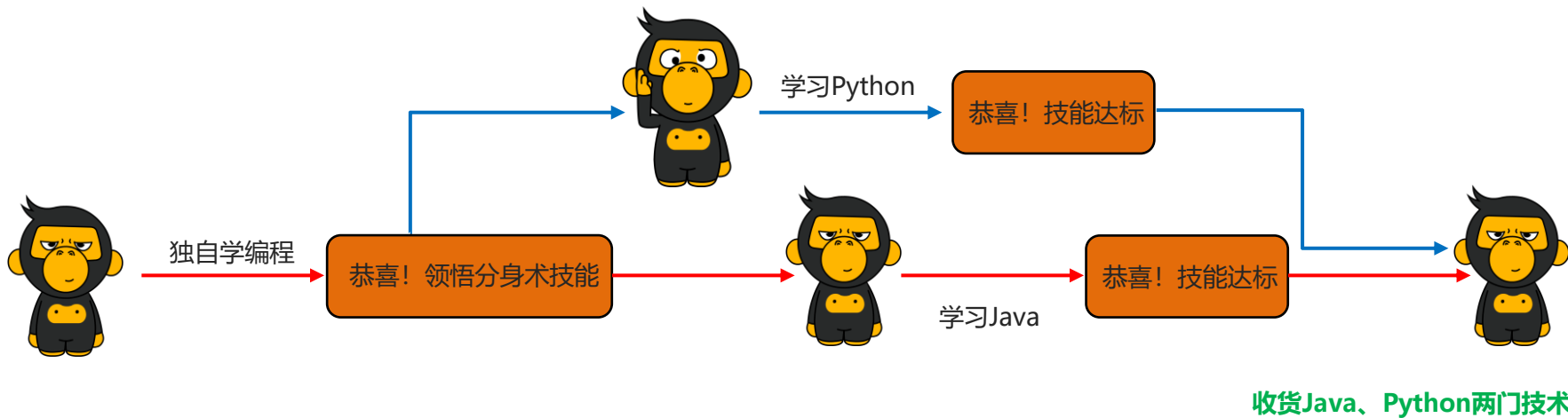
历史版本切换



历史版本切换

- 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

分支管理介绍



分支管理介绍

- 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

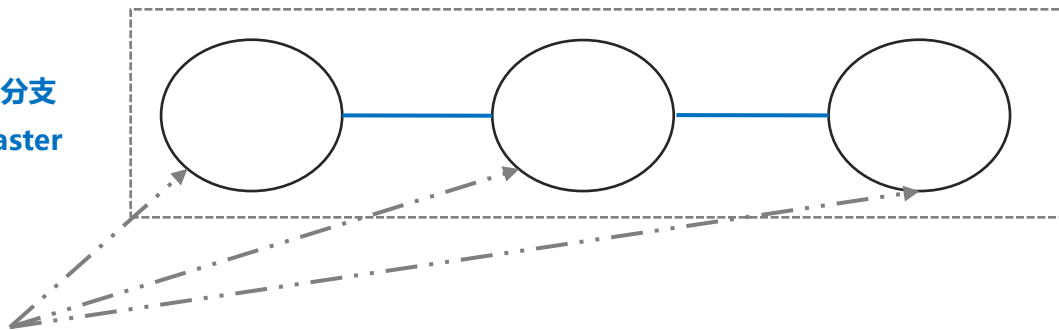
该需求应该可以使用技术手段进行实现。

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

- 分支：由每次提交的代码，串成的一条时间线

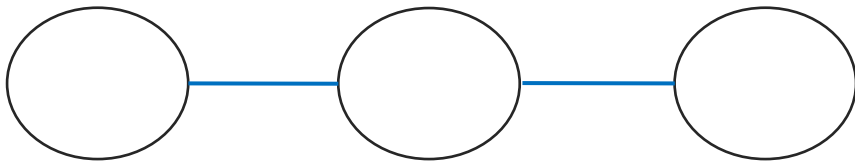
主分支
Master



分支工作流程

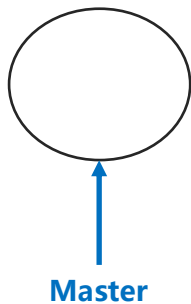
- 分支：由每次提交的代码，串成的一条时间线

主分支
Master



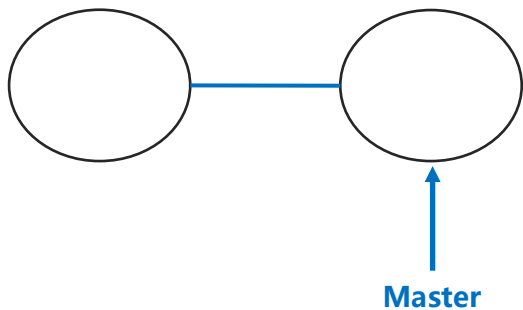
分支工作流程

- 分支：由每次提交的代码，串成的一条时间线



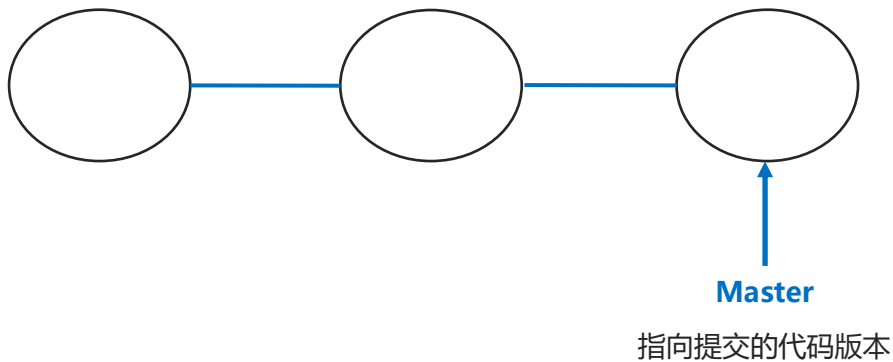
分支工作流程

- 分支：由每次提交的代码，串成的一条时间线



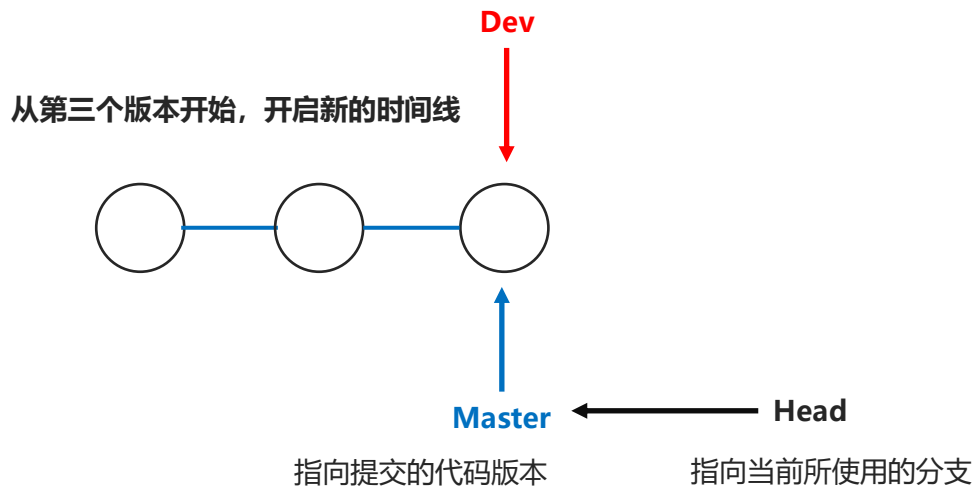
分支工作流程

- 分支：由每次提交的代码，串成的一条时间线



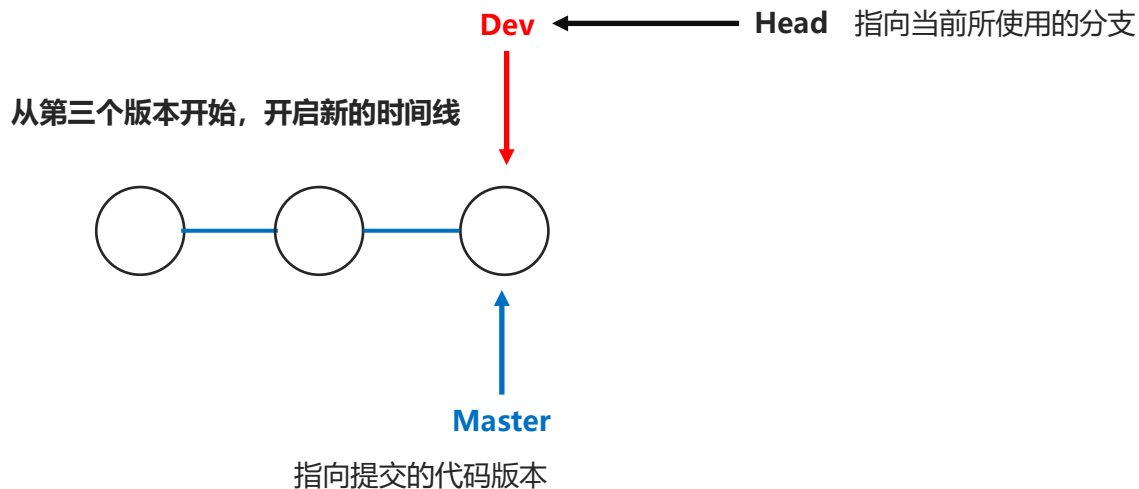
分支工作流程

- 分支：由每次提交的代码，串成的一条时间线



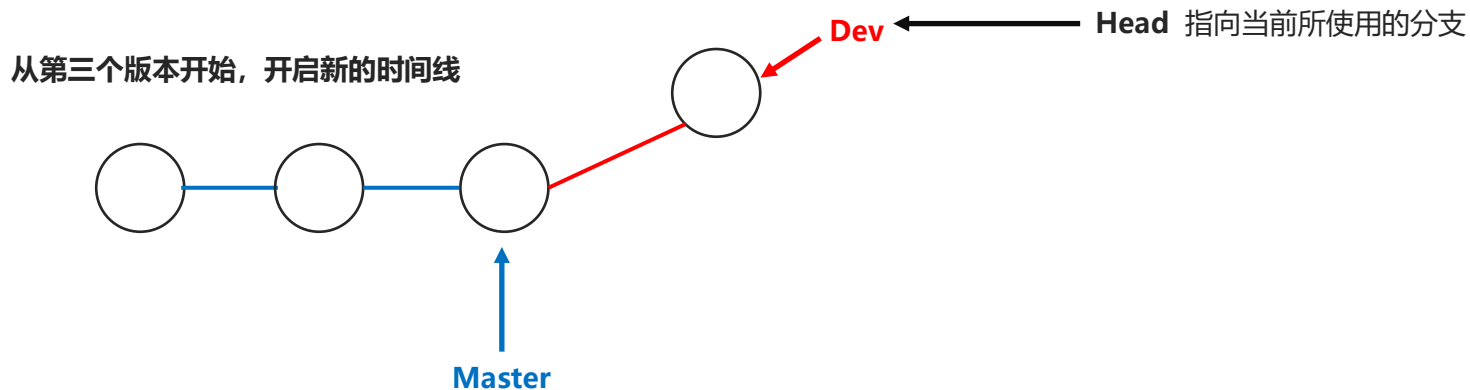
分支工作流程

- 分支：由每次提交的代码，串成的一条时间线



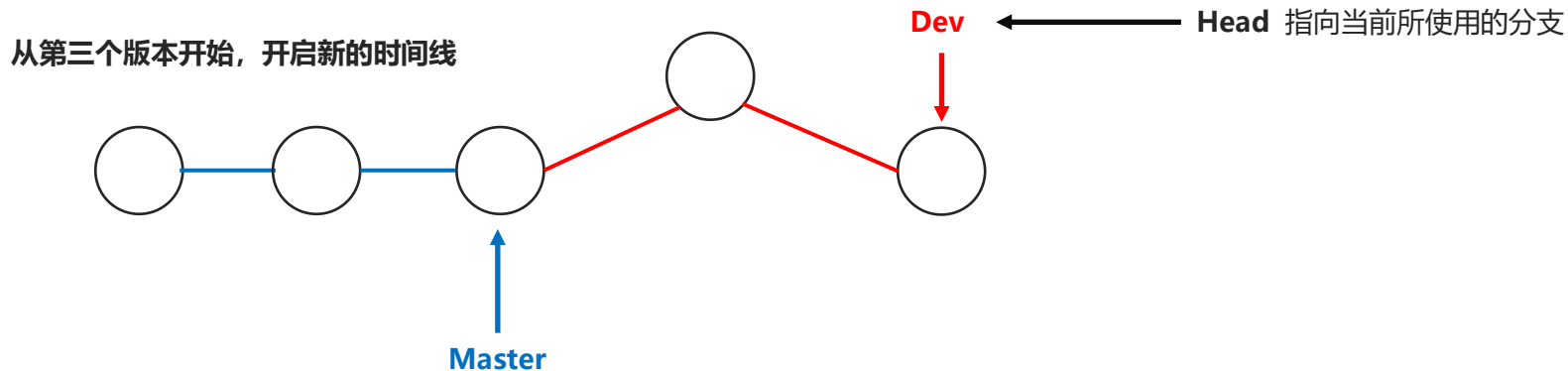
分支工作流程

- 分支：由每次提交的代码，串成的一条时间线



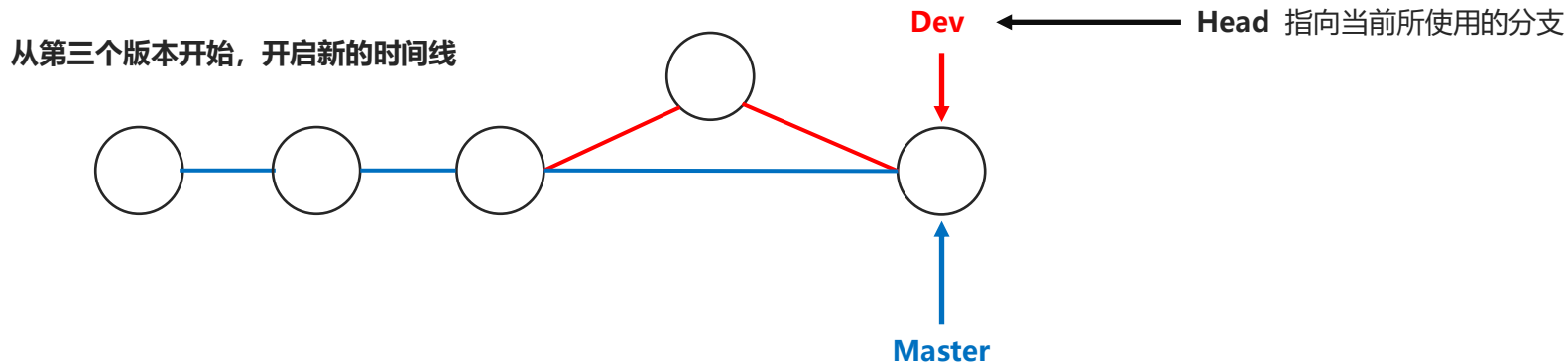
分支工作流程

- 分支：由每次提交的代码，串成的一条时间线



分支工作流程

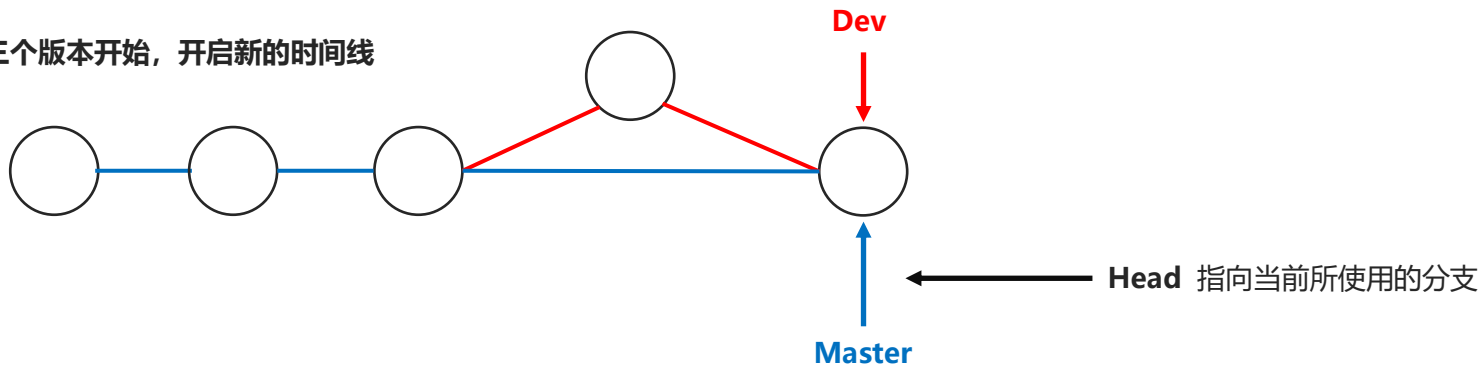
- 分支：由每次提交的代码，串成的一条时间线



分支工作流程

- 分支：由每次提交的代码，串成的一条时间线

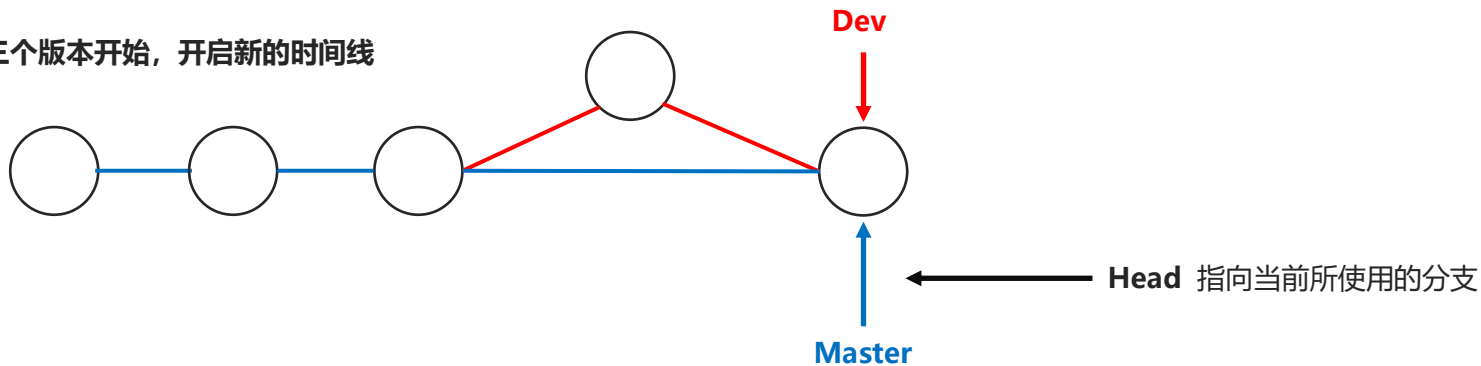
从第三个版本开始，开启新的时间线



分支工作流程

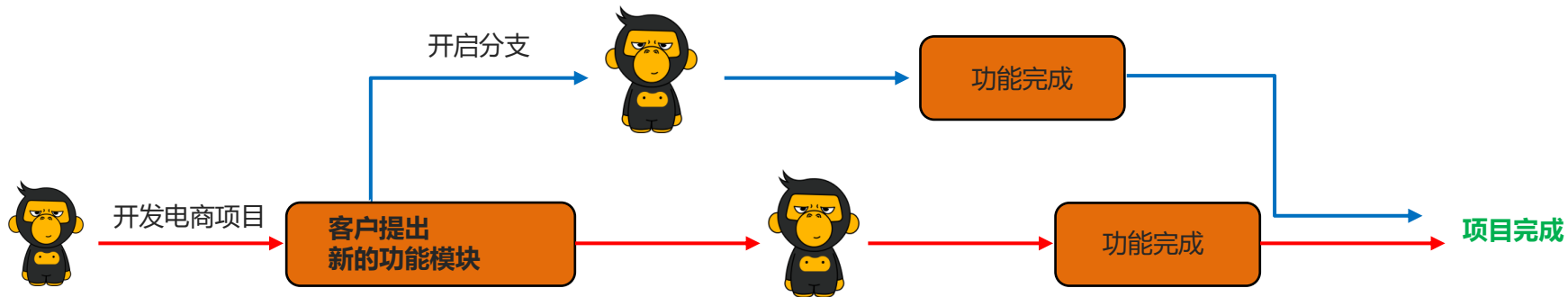
- 分支：由每次提交的代码，串成的一条时间线

从第三个版本开始，开启新的时间线



分支管理介绍

- 分支的理解



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

分支管理操作

- 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

分支管理操作

- 合并分支

合并命令: `git merge` 分支名

- 删除分支

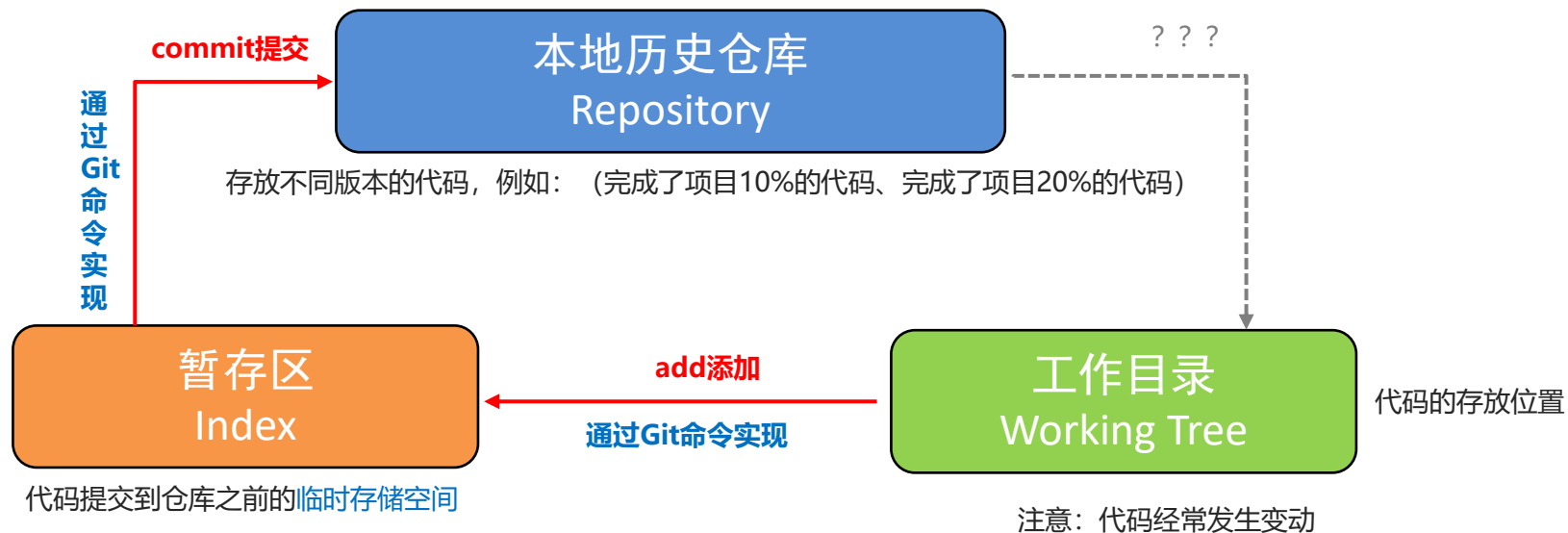
删除命令: `git branch -d` 分支名

- 查看分支列表

删除命令: `git branch`

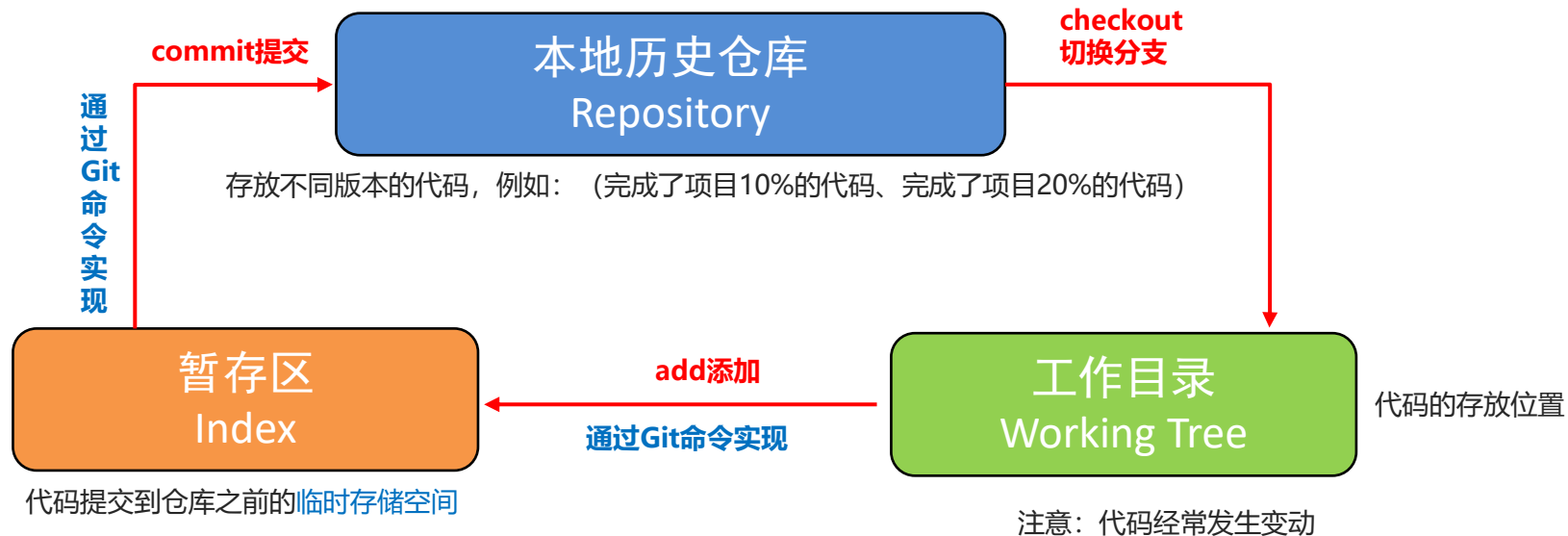
分支管理操作

- 本地仓库



分支管理操作

- 本地仓库

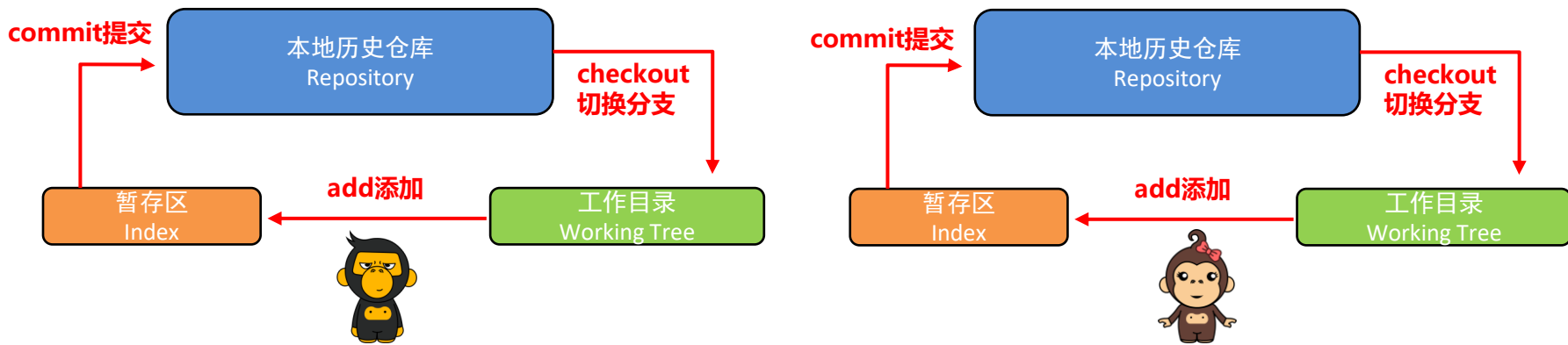


目录 Contents

- ◆ Git介绍
- ◆ Git下载和安装
- ◆ Git操作入门
- ◆ Git版本管理
- ◆ 远程仓库
- ◆ IDEA集成Git

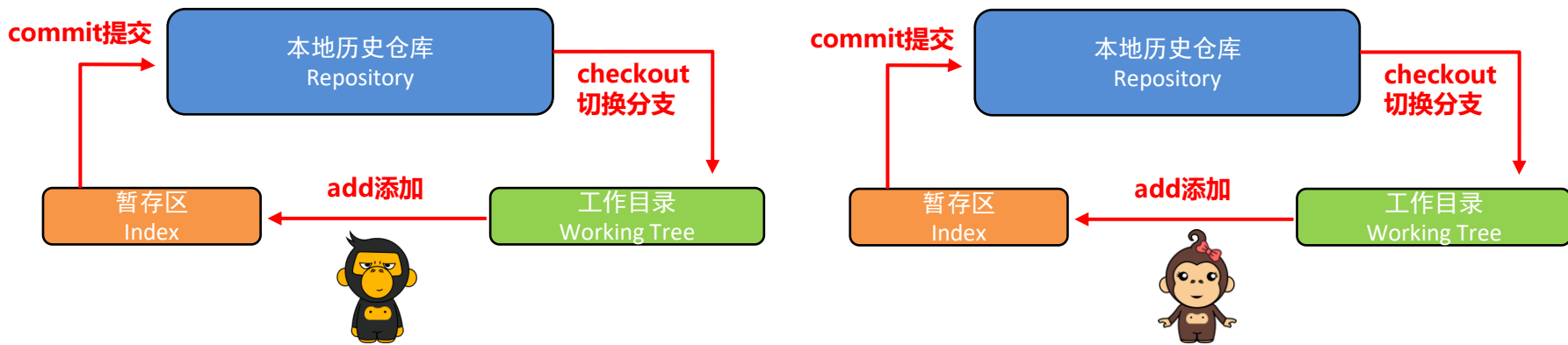
远程仓库工作流程

远程仓库
代码扩管平台（部署在公网上的一个网站）

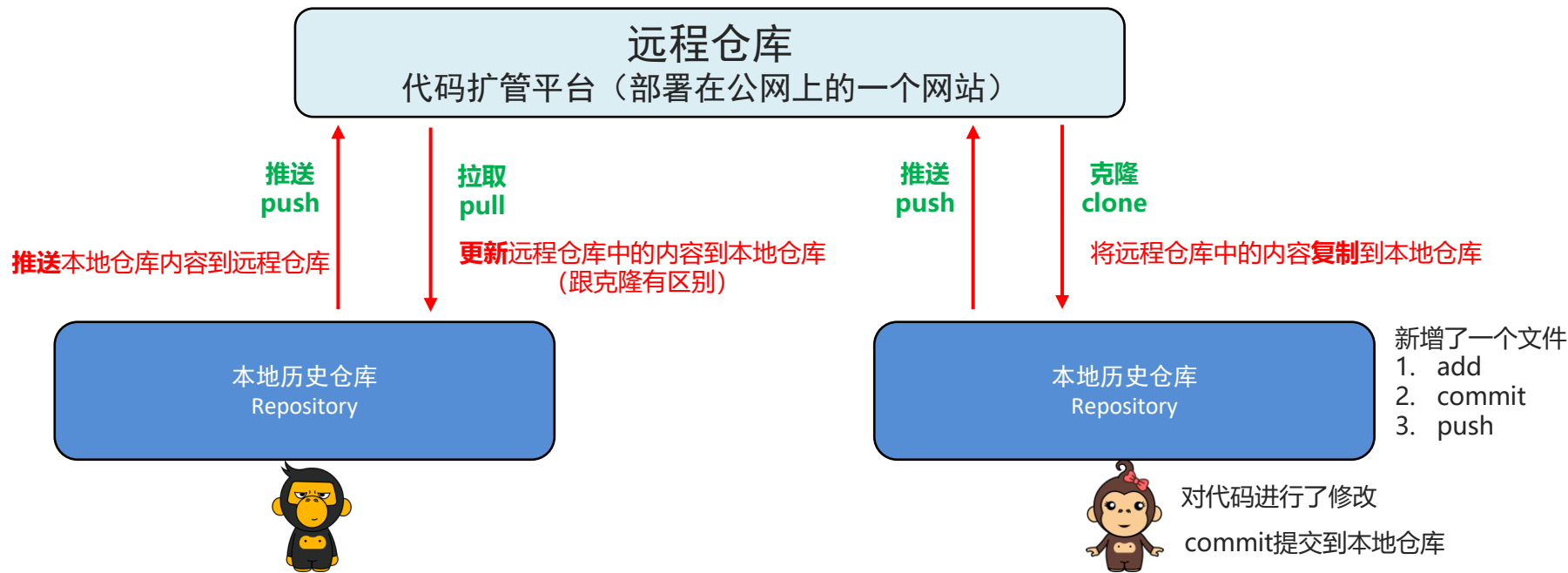


远程仓库工作流程

远程仓库
代码扩管平台（部署在公网上的一个网站）



远程仓库工作流程



远程仓库平台介绍

- 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

远程仓库平台介绍

- 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

远程仓库平台介绍

- 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

码云 开源软件 企业版 高校版 博客

搜索或跳转...

登录 注册

400万+ 开发者
600万+ 代码仓库
10万+ 企业客户
800+ 高校

稳定 · 高效 · 安全

码云 Gitee - 云端软件开发协作平台

帮助个人、团队、企业轻松实现 Git/SVN 代码托管、协作开发

加入码云 免费开通企业版

我已经有GitHub了, 为什么要用码云? 如何一键导入 GitHub 仓库?

远程仓库平台介绍



码云 Gitee

云端软件开发协作平台

李铭健 骑鹅旅行 Head of PMO

码云企业不只是一个代码托管的工具，更是带着深深 Geek 文化的项目管理平台。在码云企业版上，iGola.com 的团队协作变得更加容易，整个流程管理变得更加透明流畅。

码云企业版

hot

企业级软件开发管理平台

注册

已有帐号? [点此登录](#)

黑马Java 账号

https://gitee.com/

black_horse_java

邮箱或手机号

验证码

32 秒后可重发

..... 密码

☒ 我已阅读并同意 [使用条款](#) 以及 [非活跃帐号处理规范](#)

立即注册



使用OSChina帐号登录

其他方式登录



远程仓库平台介绍

码云

开源软件 企业版 特惠 高校版 博客 我的码云

搜索或跳转...

🔔 + 黑

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

企业

企业级软件协作开发管理 免费

- 更精细的权限管理
- 灵活、强大、可视化的项目管理
- 金融级安全策略，严守代码安全

了解更多企业版特性

组织 +

还没有组织, 立即创建

仓库 +

还没有仓库, 立即创建 或从 Github 导入

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 换一批

关注

关注

关注

关注

关注

推荐仓库 基于当前流行仓库

远程仓库平台

- 操作情况

情况1：先有本地项目，远程为空

情况2：先有远程仓库，本地为空

远程仓库平台

- 情况1：先有本地项目，远程为空
- 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

远程仓库平台

- 情况1：先有本地项目，远程为空
- 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

远程仓库平台

- 创建远程仓库



远程仓库平台

- 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

☐ 私有

☒ 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

请选择 .gitignore 模板

添加开源许可证 

请选择开源许可证

☐ 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 

☐ 使用Pull Request模板文件初始化这个仓库 

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

远程仓库平台

- 创建远程仓库

注意：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.../repo1/'
```

远程仓库平台

- 创建远程仓库
- 生成SSH公钥步骤
 1. 设置Git账户
 2. 生成SSH公钥
 3. 设置账户公钥
 4. 公钥测试

远程仓库平台

- 创建远程仓库

- 生成SSH公钥步骤

1. 设置Git账户

- 命令

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"

(设置全局账户名和邮箱)

git config --global user.email "邮箱"

cd ~/.ssh

(查看是否生成过SSH公钥)

```
MINGW64:/d/my_project
admin@LAPTOP-L04AUSV0 MINGW64 /d/my_project (master)
$ git config user.name
heimaTest
查看用户名和邮箱
admin@LAPTOP-L04AUSV0 MINGW64 /d/my_project (master)
$ git config user.email
heimaTest@itcast.cn
admin@LAPTOP-L04AUSV0 MINGW64 /d/my_project (master)
$ git config --global user.name "heimaTest"
--global: 表示这台机器上所有的Git仓库都会使用这个配置
admin@LAPTOP-L04AUSV0 MINGW64 /d/my_project (master)
$ git config --global user.email "heimaTest@itcast.cn"
admin@LAPTOP-L04AUSV0 MINGW64 /d/my_project (master)
$ cd ~/.ssh
bash: cd: /c/Users/haoyu/.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)
```

远程仓库平台

- 创建远程仓库
- 生成SSH公钥步骤
 1. 设置Git账户
 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
admin@APTOP-I04AUSV0: MINGW64 /d/my_project (master)
$ 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]-----+
```


远程仓库平台

- 创建远程仓库
- 生成SSH公钥步骤
 1. 设置Git账户
 2. 生成SSH公钥

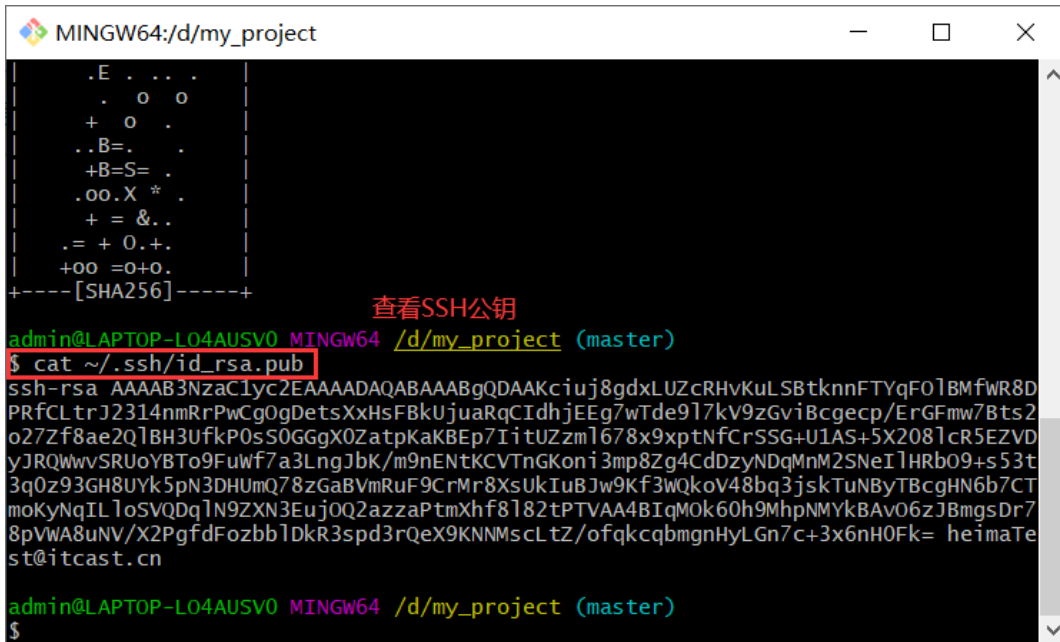
- 生成命令

`ssh-keygen -t rsa -C "邮箱"`

- 注意：这里需要敲3次回车

- 查看命令

`cat ~/.ssh/id_rsa.pub`



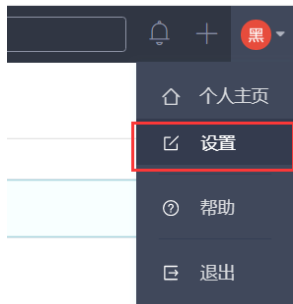
```
MINGW64:/d/my_project
.E . . .
. o o
+ o .
..B=.
.+B=S=
.oo.X *.
+ = &..
.= + O.+
+oo =o+o.
+-----[SHA256]-----+

admin@LAPTOP-L04AUSV0 MINGW64 /d/my_project (master)
$ cat ~/.ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQDAACKcuj8gdxLUZcRHvKuLSBtknnFTYqFOlBMfWR8D
PRfCLtrJ2314nmRrPwCgOgDetsXxHsFBkUjuaRqCiDhjEEg7wTde9l7kV9zGviBcgecp/ErGFmw7Bts2
o27Zf8ae2QlBH3UfkP0sS0GGgX0ZatpKaKBEp7IitUZzm1678x9xptNfCrSSG+U1AS+5X208lcr5EZVD
yJRQWwvSRUoYBTo9FuWf7a3LNgJbK/m9nEntKCVTnGKoni3mp8Zg4CdDzyNDqMnM2SNeIlHRbO9+s53t
3q0z93GH8UYk5pN3DHUmQ78zGaBVMruF9CrMr8XsUkIuBjw9Kf3wQkoV48bq3jskTuNByTBcgHN6b7CT
moKyNqILloSVQDqlN9ZXN3Euj0Q2azzaPtmXhf8l82tPTVAA4BIqM0k60h9MhpNMYkBAvO6zJBmgsDr7
8pVWA8uNV/X2PgfdFozbb1DkR3spd3rQeX9KNNMscLtz/ofqkcqbmgNHyLGN7c+3x6nH0Fk= heimaTe
st@itcast.cn

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

远程仓库平台

- 创建远程仓库
- 步骤
 1. 设置Git账户
 2. 生成SSH公钥
 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码云通讯的时候使用安全连接 (Git的Remote要使用SSH地址)

您当前的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 AAAAB3NzaC1yc2EAAAADAQABAAQgQDAACiuj8gdxLUZcRHvKulSBtknnFTYqFOlBMWR8DPRfCLtrJ2314nmRrPwCgOgDetsXhSfBkUjuaRqCldhJEg7wTde9l7kV9zGvIBcgcp/ErGFmw7Bts2o27Zf8ae2QlBH3UfrP0s50GGgX0ZatpKaKBEP7litUZzmI678x9ptNfCrSSG+UJAS+5X208lcR5EZVDyIRQWwwSRUoYBTto9FuWl7a3LngJbK/m9nENtKCVTnGKoni3mp8Zq4CdDzyNDqMnM2SNellHRbOQ+s53t3q0z93GH8UYk5pN3DHUmQ78zGaBvmRuF9CrMr8xsUktuBjw9KF3WQkoV48bq3jskTuNBylT8cgHN6b7CTmoKyNqILloSVQDqIN9ZN3EujOQ2azzaPtmXhf8l82lPTVA48lqMOK60h9MhpNMYkBAvO6zJBmgsDr78pVWA8uNV/X2PafdfFozbblDkR3spd3rQeX9KNNMscLtz/afakcqbmqnHyLgnZc+3x6nH0Fk= heimaTest@itcast.cn
```

确定

远程仓库平台

- 创建远程仓库

- 步骤

1. 设置Git账户
2. 生成SSH公钥
3. 设置账户公钥
4. 公钥测试

- 命令

`ssh -T git@gitee.com`

```
MINGW64:/d/my_project
admin@LAPTOP-LO4AUSV0 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/eye1w8icdBgrQp+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-LO4AUSV0 MINGW64 /d/my_project (master)
$
```

远程仓库平台

- 推送到远程仓库

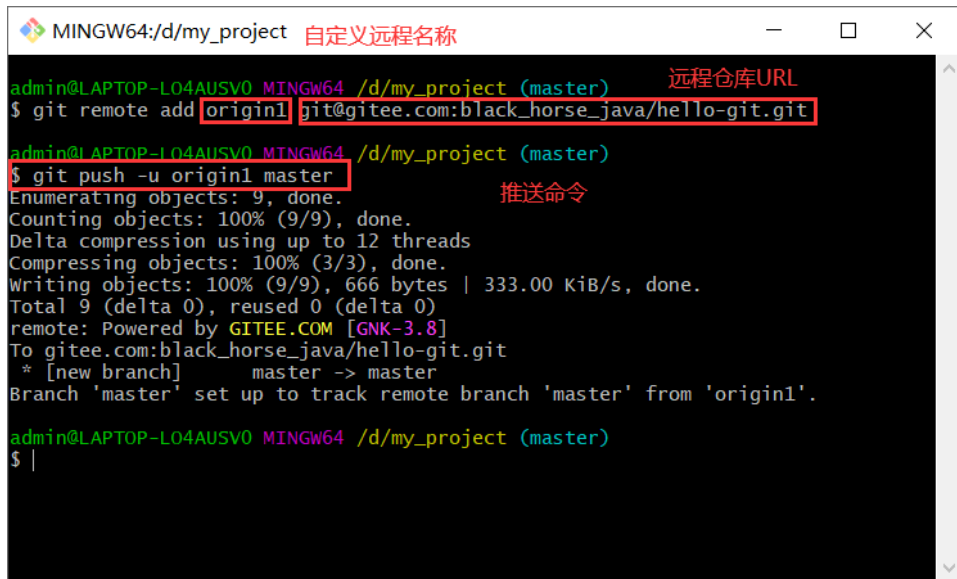
- 步骤

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`



```
MINGW64:/d/my_project 自定义远程名称
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)
$ |
```

远程仓库平台

- 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`



远程仓库平台

- 操作情况

情况1：先有本地项目，远程为空

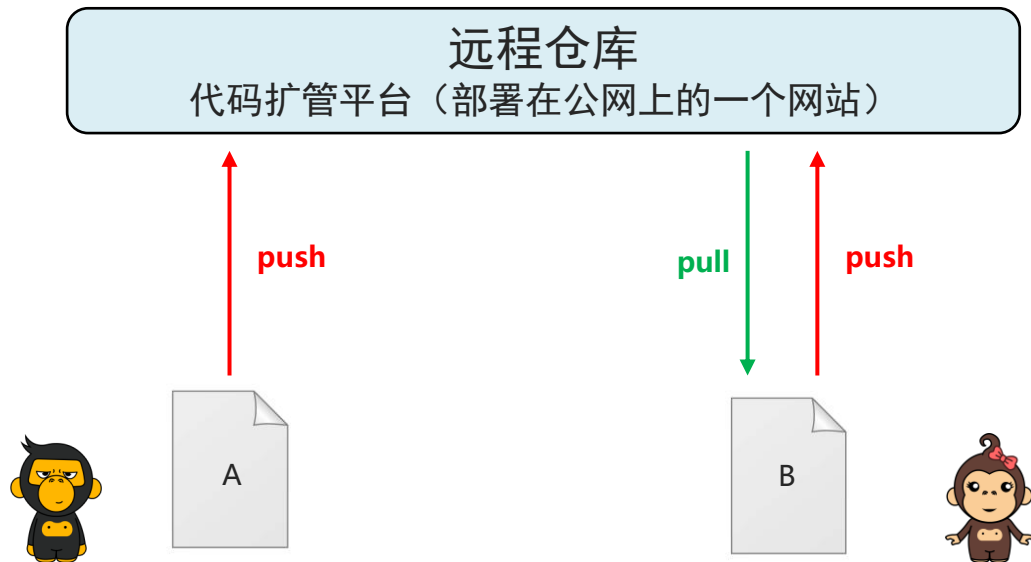
情况2：先有远程仓库，本地为空

远程仓库平台

- 情况2：先有远程仓库，本地为空
- 步骤：
 1. 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
 2. 创建新文件，添加并提交到本地仓库
 3. 推送至远程仓库
 4. 项目拉取更新
拉取命令：git pull 远程仓库名 分支名

远程仓库平台

- 注意：代码冲突



远程仓库平台

- 注意：代码冲突

```
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.
```

远程仓库平台

- 注意：代码冲突

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)
$ |
```

远程仓库平台

- 注意：代码冲突

```
admin@LAPTOP-L04AUSV0 MINGW64 /d/my_project (master|MERGING)
$ cat test.txt
<<<<<< HEAD
update count=10
=====
update count=100
>>>>>> 83fa5777883df2f666807e9a8fa2041bf8dc70b8
```

<<<<<<和>>>>>>中间的内容,就是冲突部分

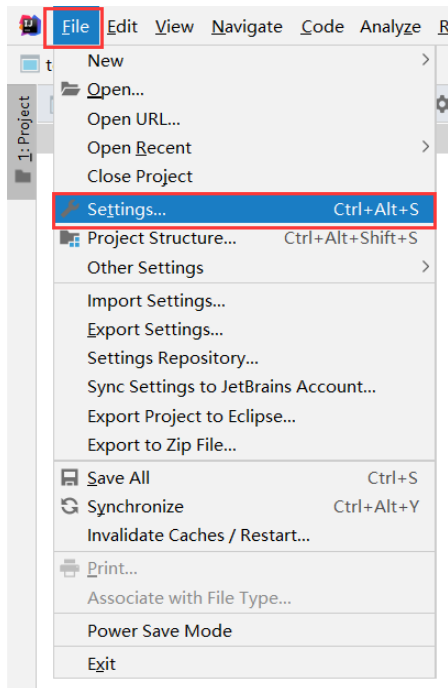
1. 修改冲突行，保存，即可解决冲突。
2. 重新add冲突文件并commit到本地仓库，重新push到远程

目录 Contents

- ◆ Git介绍
- ◆ Git下载和安装
- ◆ Git操作入门
- ◆ Git版本管理
- ◆ 远程仓库
- ◆ IDEA集成Git

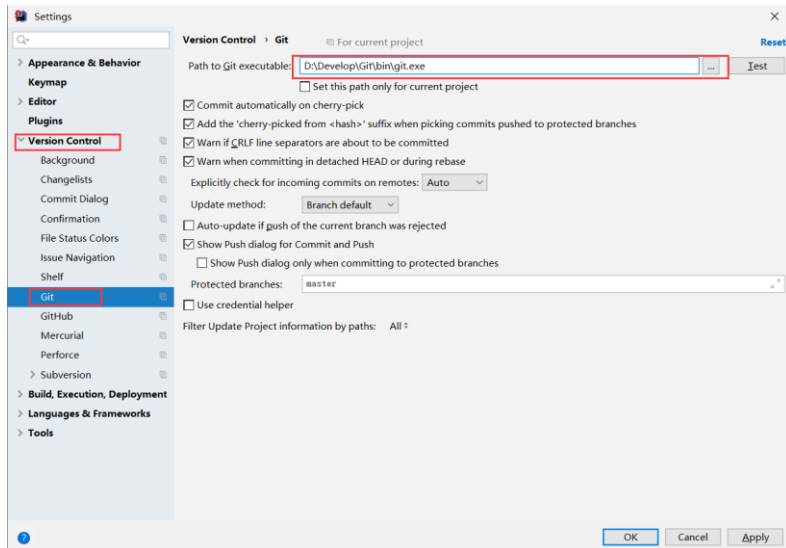
IDEA中配置Git

1. File --- Settings



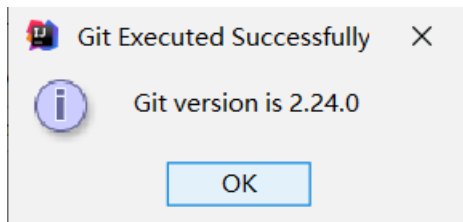
IDEA中配置Git

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



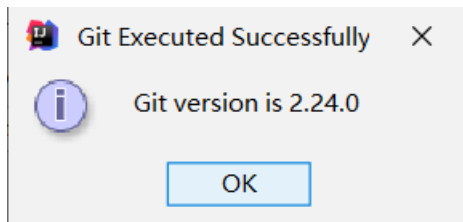
IDEA中配置Git

3. 点击 Test 测试



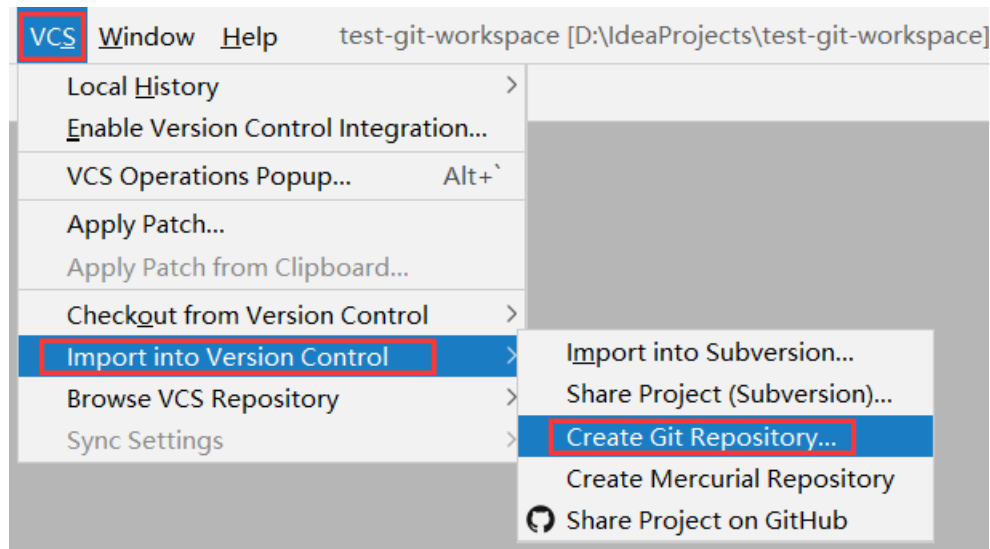
IDEA中配置Git

3. 点击 Test 测试



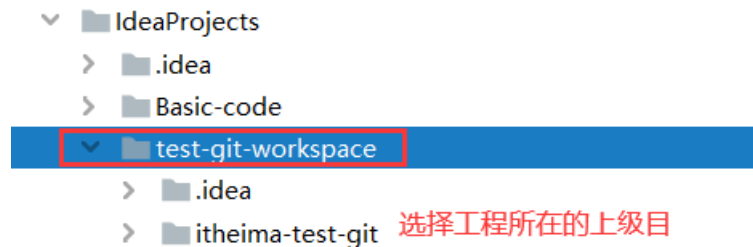
本地仓库操作

- 提交本地项目 - 创建本地仓库



本地仓库操作

- 提交本地项目 - 创建本地仓库

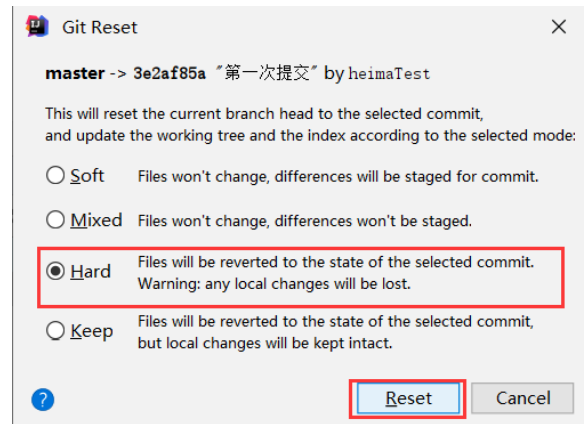
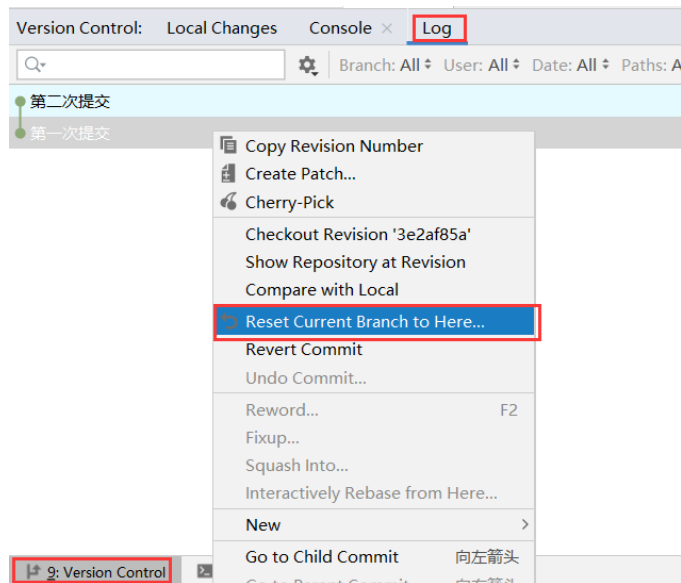


本地仓库操作

- 提交本地项目 - 设置忽略

本地仓库操作

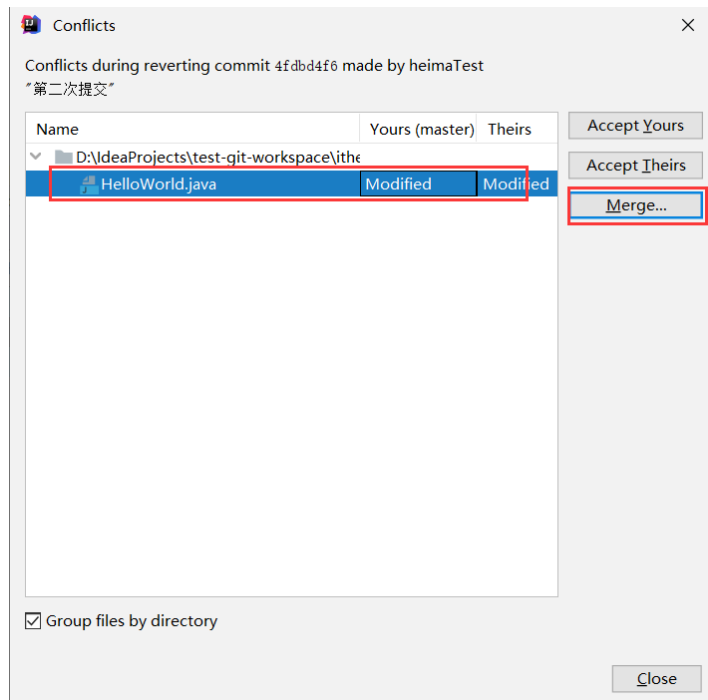
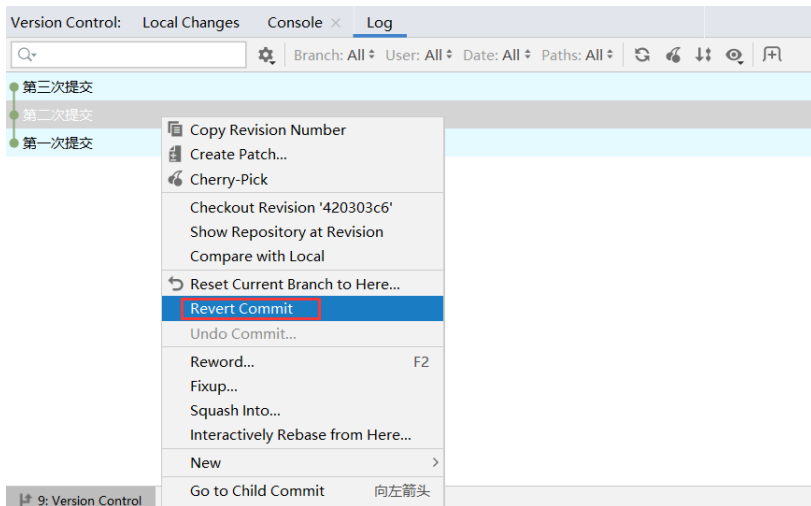
- 提交本地项目 - 版本切换



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

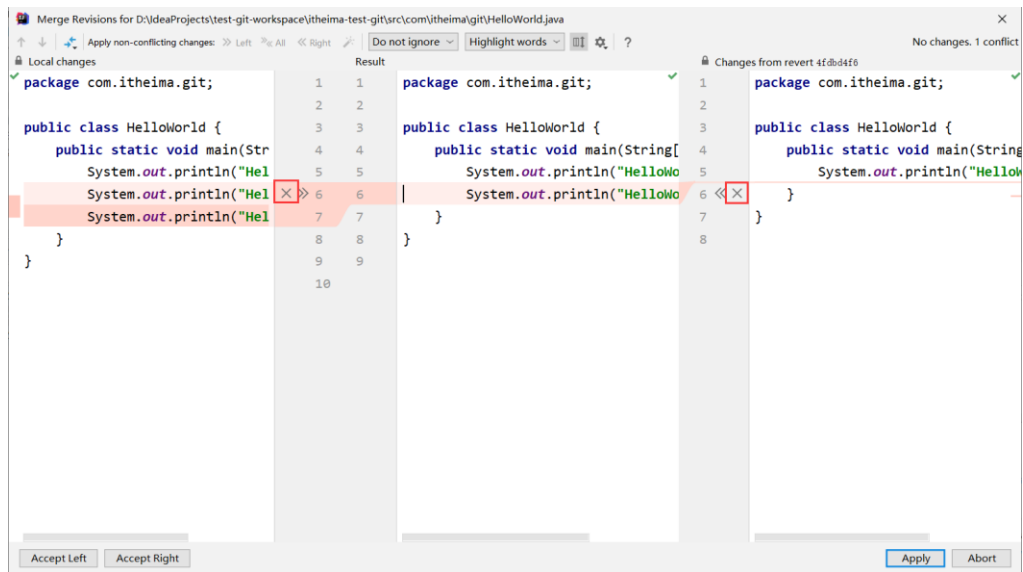
本地仓库操作

- 提交本地项目 - 版本切换



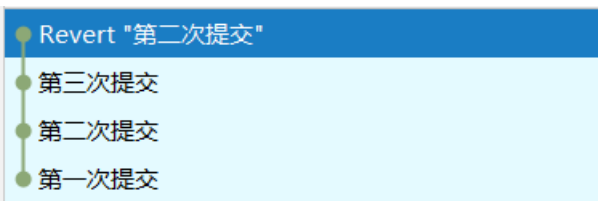
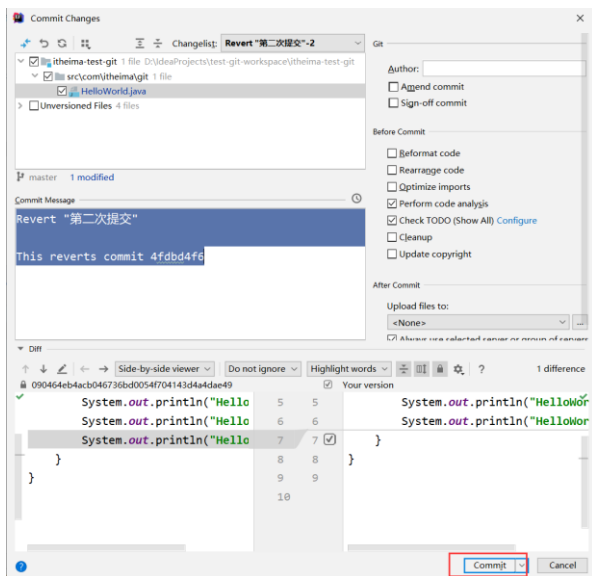
本地仓库操作

- 提交本地项目 - 版本切换



本地仓库操作

- 提交本地项目 - 版本切换

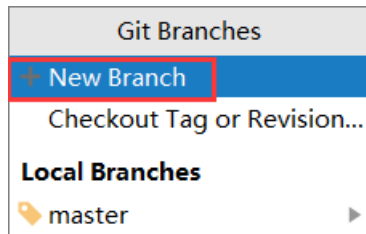
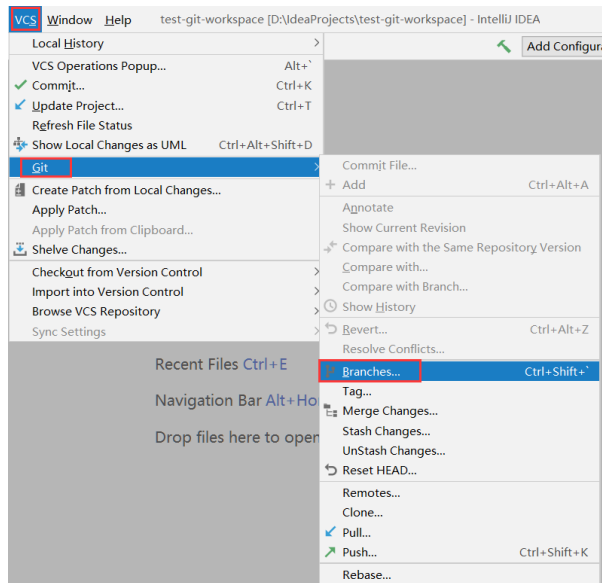


Revert 操作会当成一个新的提交记录

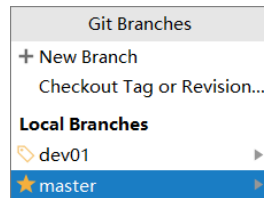
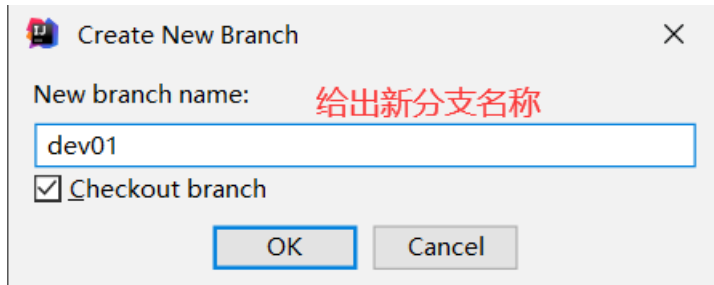
这种回退的好处在于，如果后悔了“回退”这个操作
也可以回退到没有回退之前的版本
因为历史记录还保留提交记录

本地仓库操作

- 分支管理 - 创建并切换分支

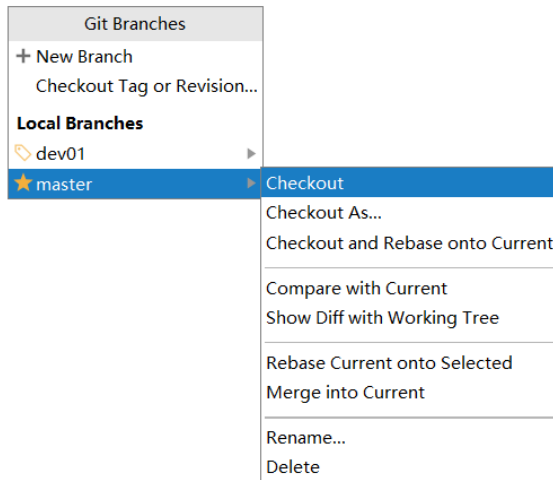
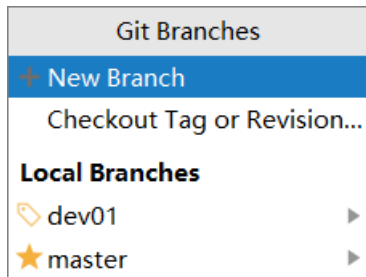


如果勾选了Checkout branch
代表创建后，会自动切换



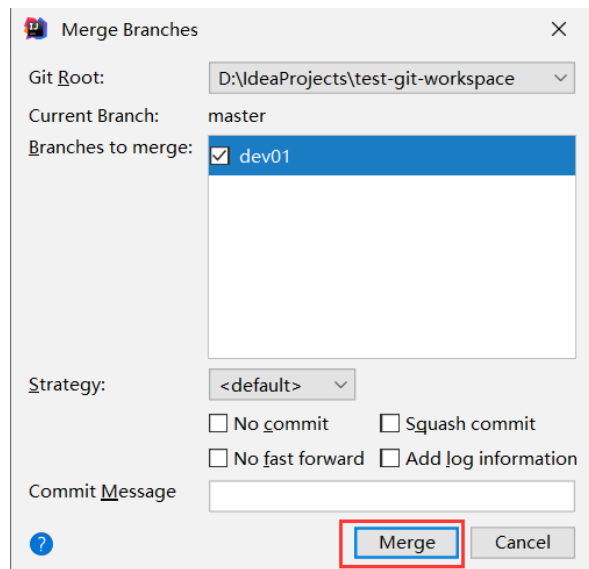
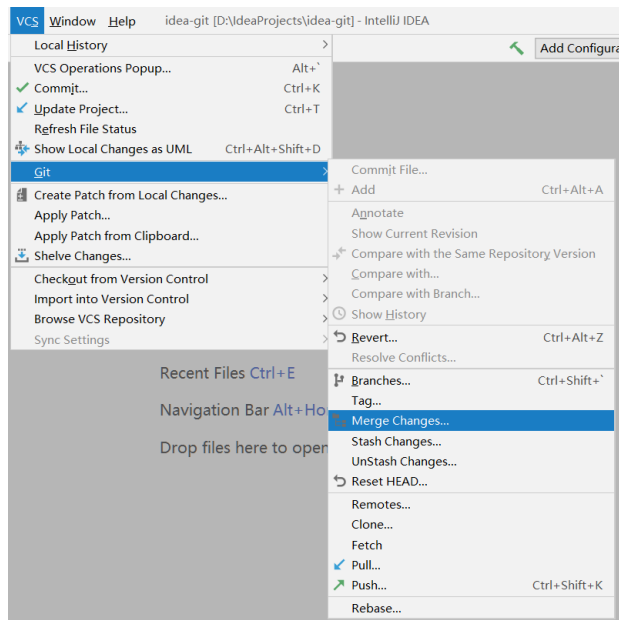
本地仓库操作

- 分支管理 - 创建并切换分支



本地仓库操作

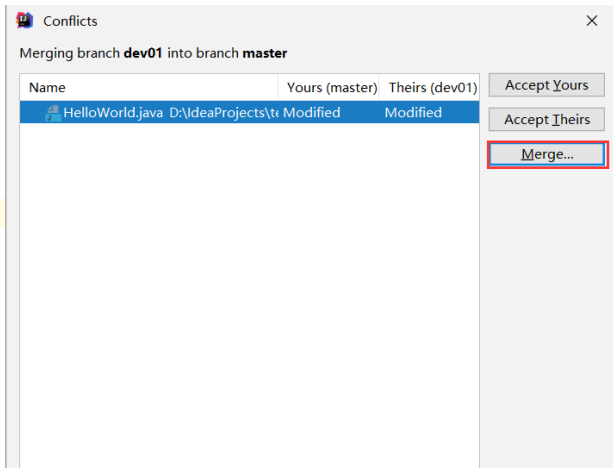
● 分支管理 – 合并分支



本地仓库操作

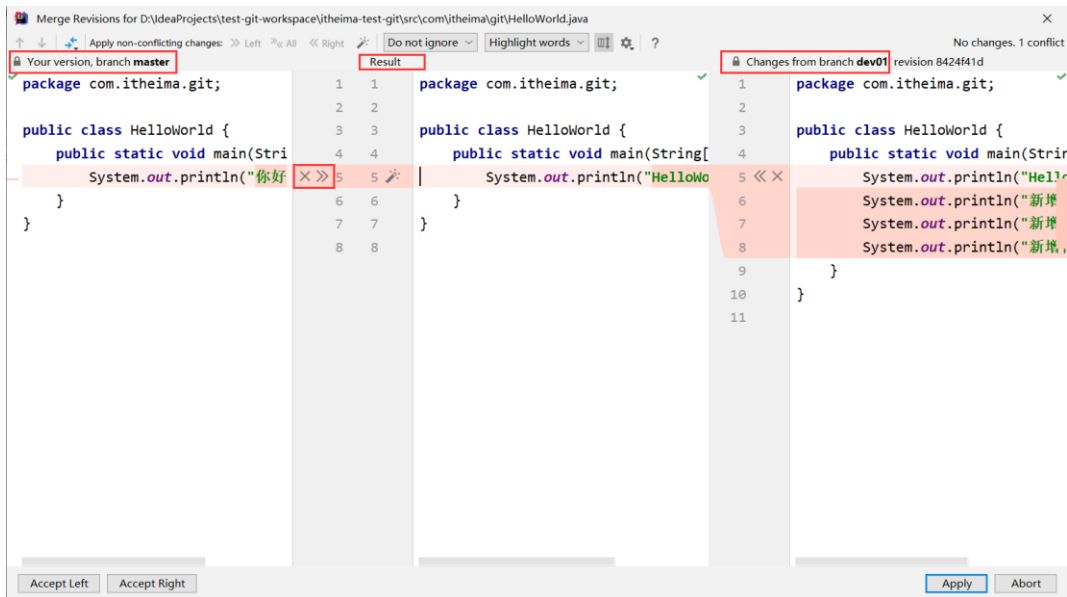
- 分支管理 – 合并分支代码冲突

```
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  
    }  
}
```



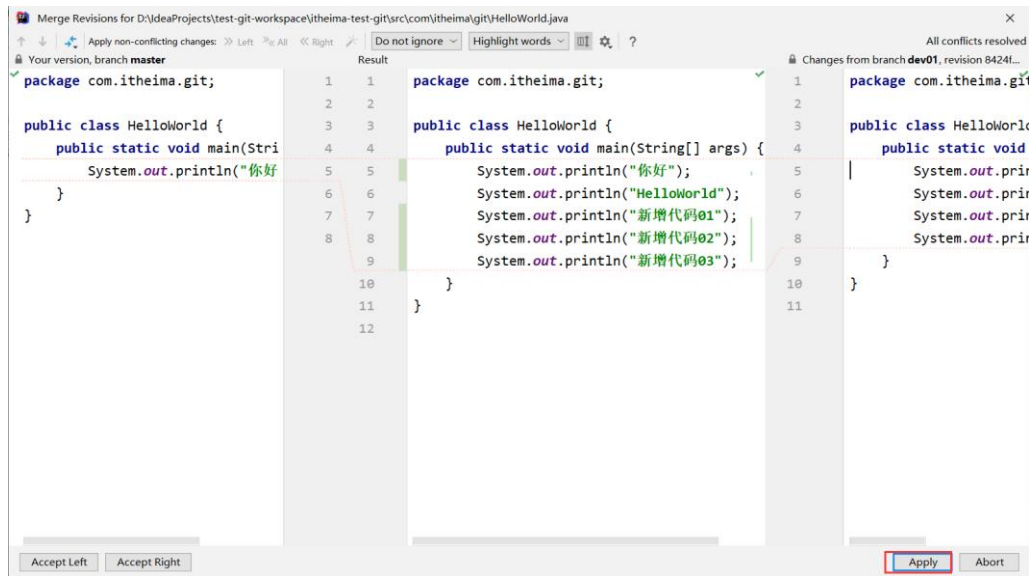
本地仓库操作

- 分支管理 – 合并分支



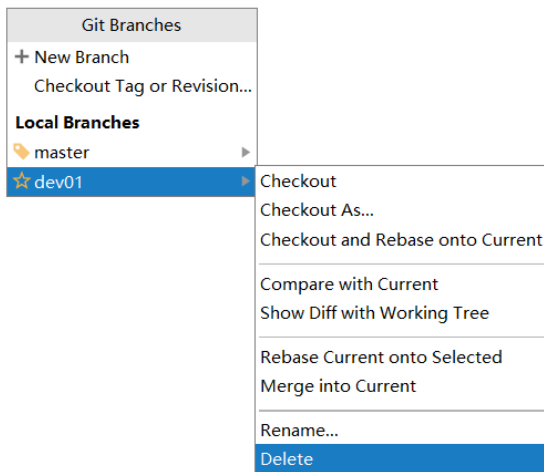
本地仓库操作

- 分支管理 – 合并分支



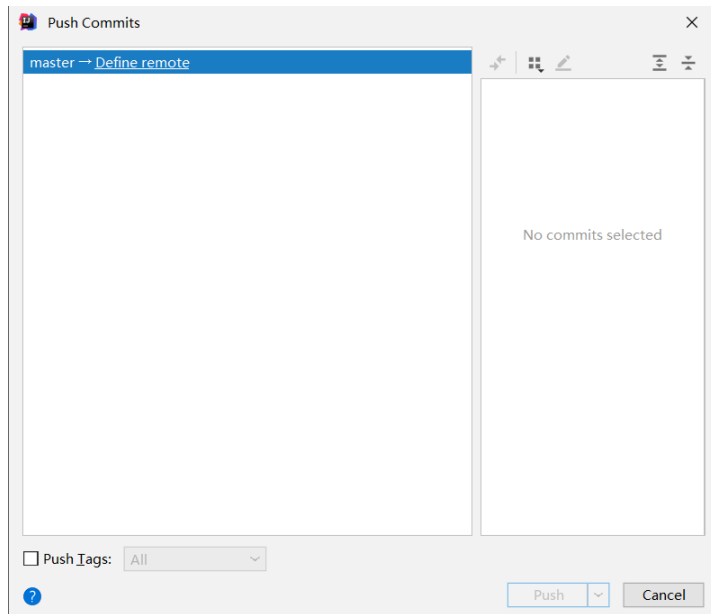
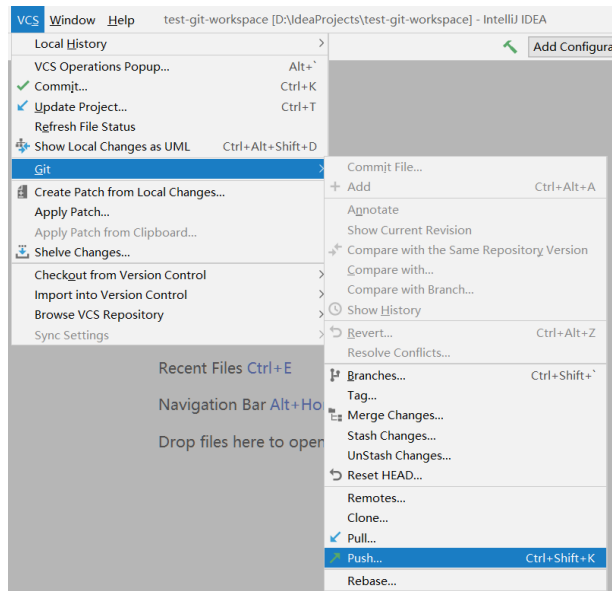
本地仓库操作

- 分支管理 – 删除分支



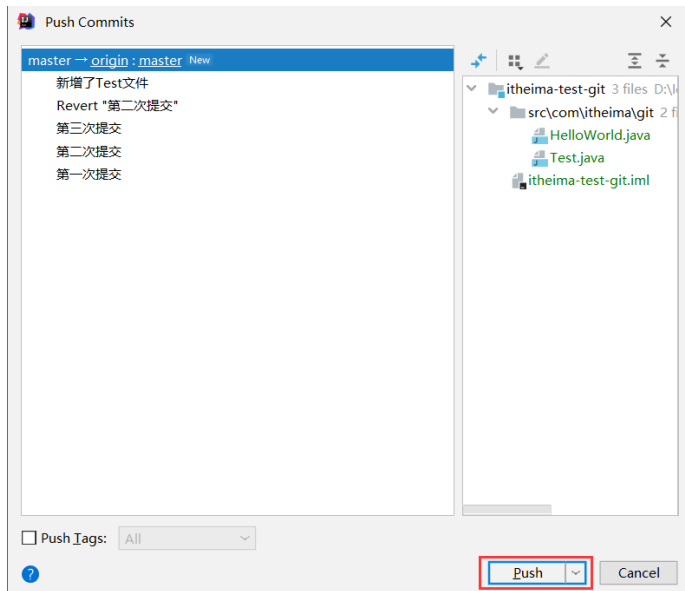
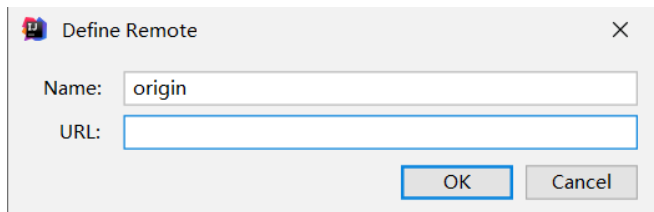
远程仓库操作

- 本地推送到远程



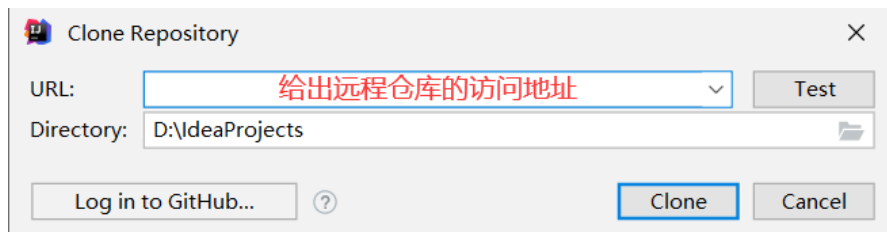
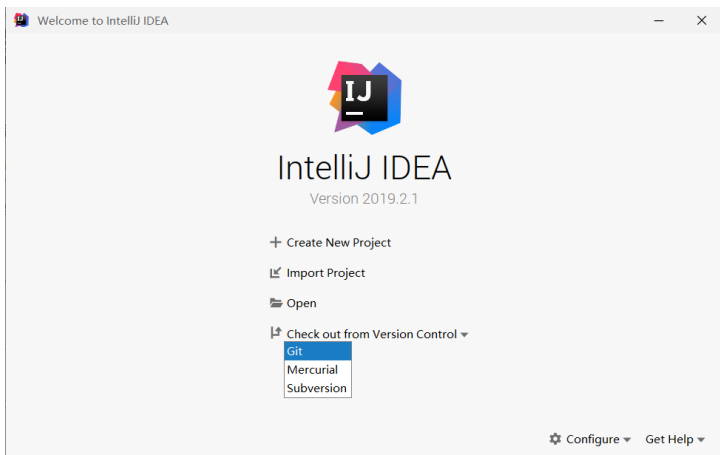
远程仓库操作

- 本地推送到远程



远程仓库操作

- 远程克隆到本地





传智播客旗下高端IT教育品牌