

Beijing Forest Studio
北京理工大学信息系统及安全对抗实验中心



多人协作开发-Git使用介绍

硕士研究生 李嘉玮

2023年02月05日

- 背景介绍
- 基本概念
- Git使用
- 总结
- 参考文献

- 预期收获
 - 了解版本控制的基本概念和不同版本控制的区别
 - 熟悉Git底层工作原理
 - 掌握Git多人协作流程及核心用法



背景介绍

- 多人协作开发
 - 多人协作：负责人和开发人员
 - 开发：满足客户需求并且解决客户问题
 - 开发流程
 - 需求分析与设计
 - 需求实现
 - 编码、测试、部署
 - 多人高效合作
 - 验收及维护
 - v1.0、v2.0……
- Peach项目中的多人__开发
 - “孤立式” 开发

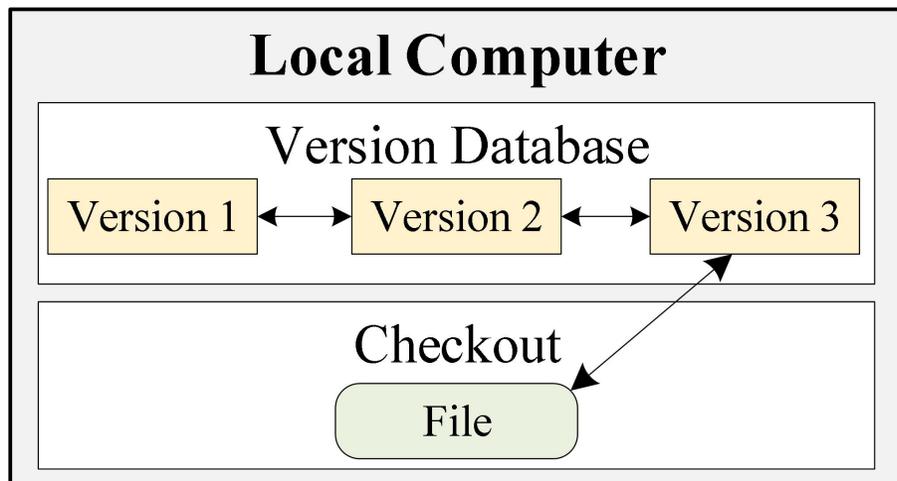


- 版本控制系统

- 一种记录一个或若干文件内容**变化**，以便查阅特定版本修订情况的系统

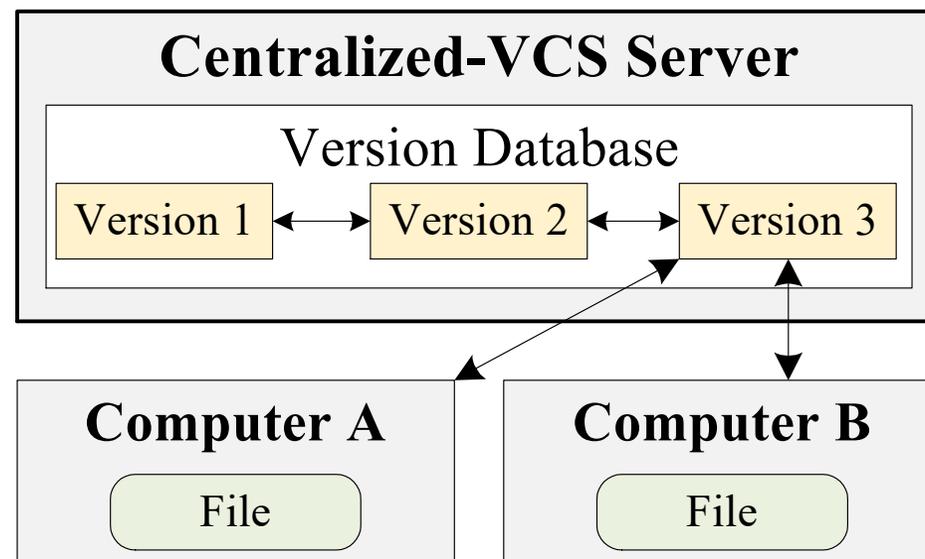
- 本地版本控制系统

- 在本地计算机的磁盘上保持补丁集
- 缺点：无法实现协同办公，**效率低**



- 集中式版本控制系统

- 版本库存集中在中央服务器上，所有开发人员必须连接到服务器上，实现协同开发
- 缺点：**单点故障**



- 分布式版本控制系统

- 客户端不仅只提取最新版本的文件快照，并且获取代码仓库完整的镜像内容

- 特点

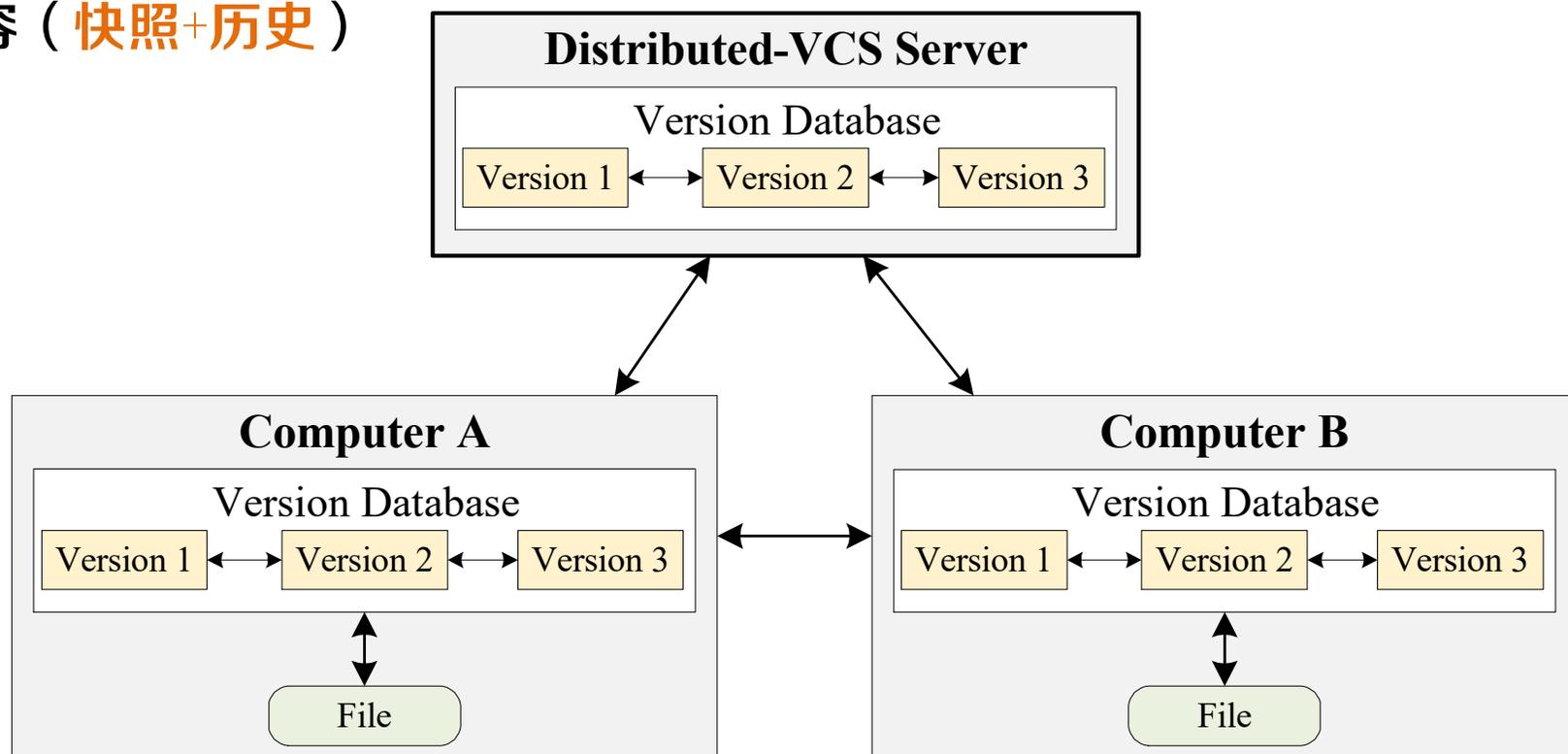
- 所有终端保存全部内容（快照+历史）
 - 支持离线管理
 - 支持非中心化管理

- 优点

- 速度快
 - 稳定性高
 - 可扩展性强

- 工具

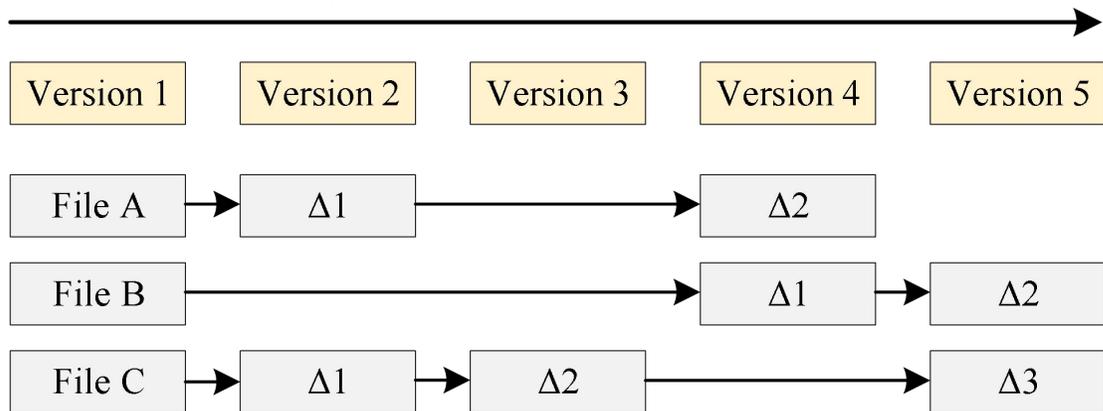
- Git
 - Bazaar



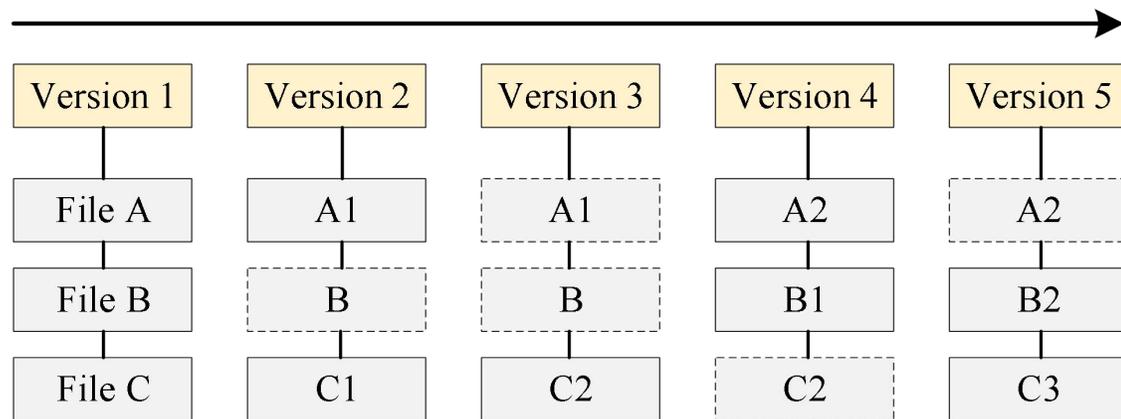
- 基于差异 (Delta-based) 的数据存储方式
 - 将数据存储为对每个文件的基本版本的更改
- Git
 - 开源的**分布式**版本控制系统，最初被用于Linux内核开发
 - **数据即快照流**：将数据存储为项目的快照
 - Git直接记录完整快照，而非差异比较



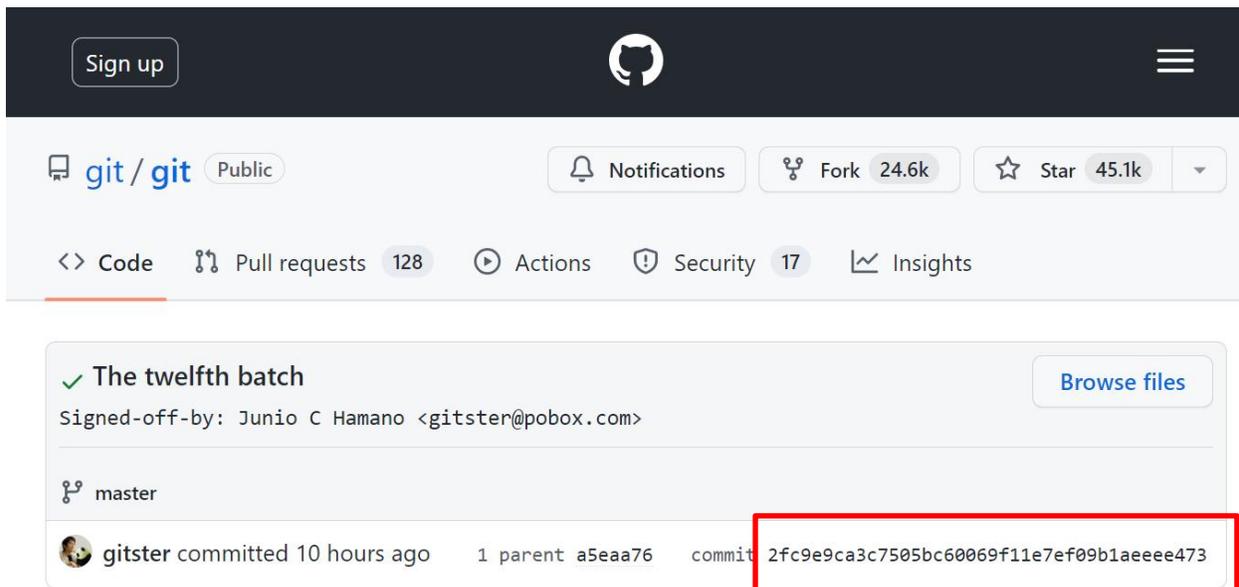
Variation With Time



Variation With Time



- Git使用SHA-1进行文件校验和文件索引



Sign up

git / git Public

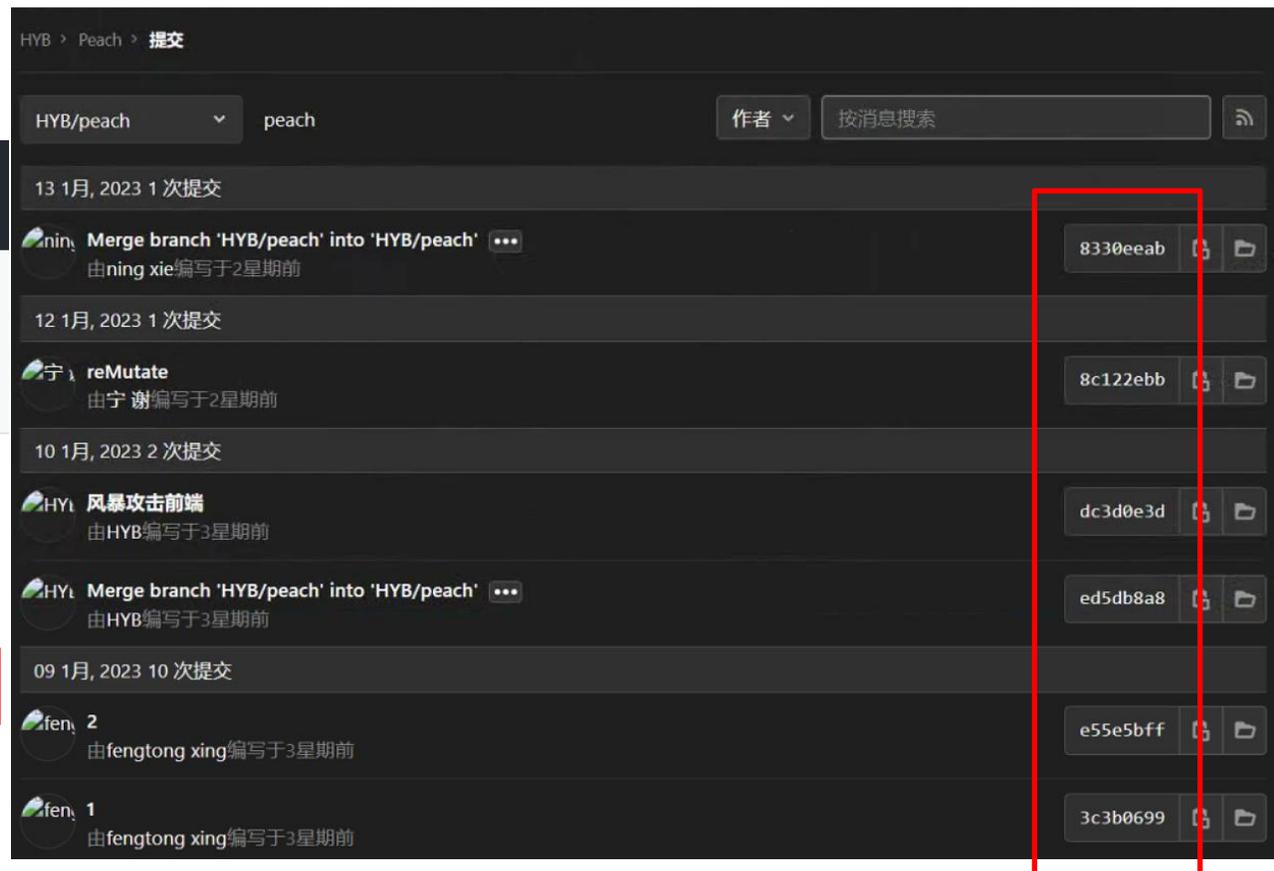
Code Pull requests 128 Actions Security 17 Insights

✓ The twelfth batch [Browse files](#)

Signed-off-by: Junio C Hamano <gitster@pobox.com>

master

gitster committed 10 hours ago 1 parent a5eaa76 commit 2fc9e9ca3c7505bc60069f11e7ef09b1aeeee473



HYB / peach 提交

HYB/peach peach 作者 按消息搜索

13 1月, 2023 1次提交

Merge branch 'HYB/peach' into 'HYB/peach' ...
由ning xie编写于2星期前 8330eeab

12 1月, 2023 1次提交

reMutate
由宁 谢编写于2星期前 8c122ebb

10 1月, 2023 2次提交

风暴攻击前端
由HYB编写于3星期前 dc3d0e3d

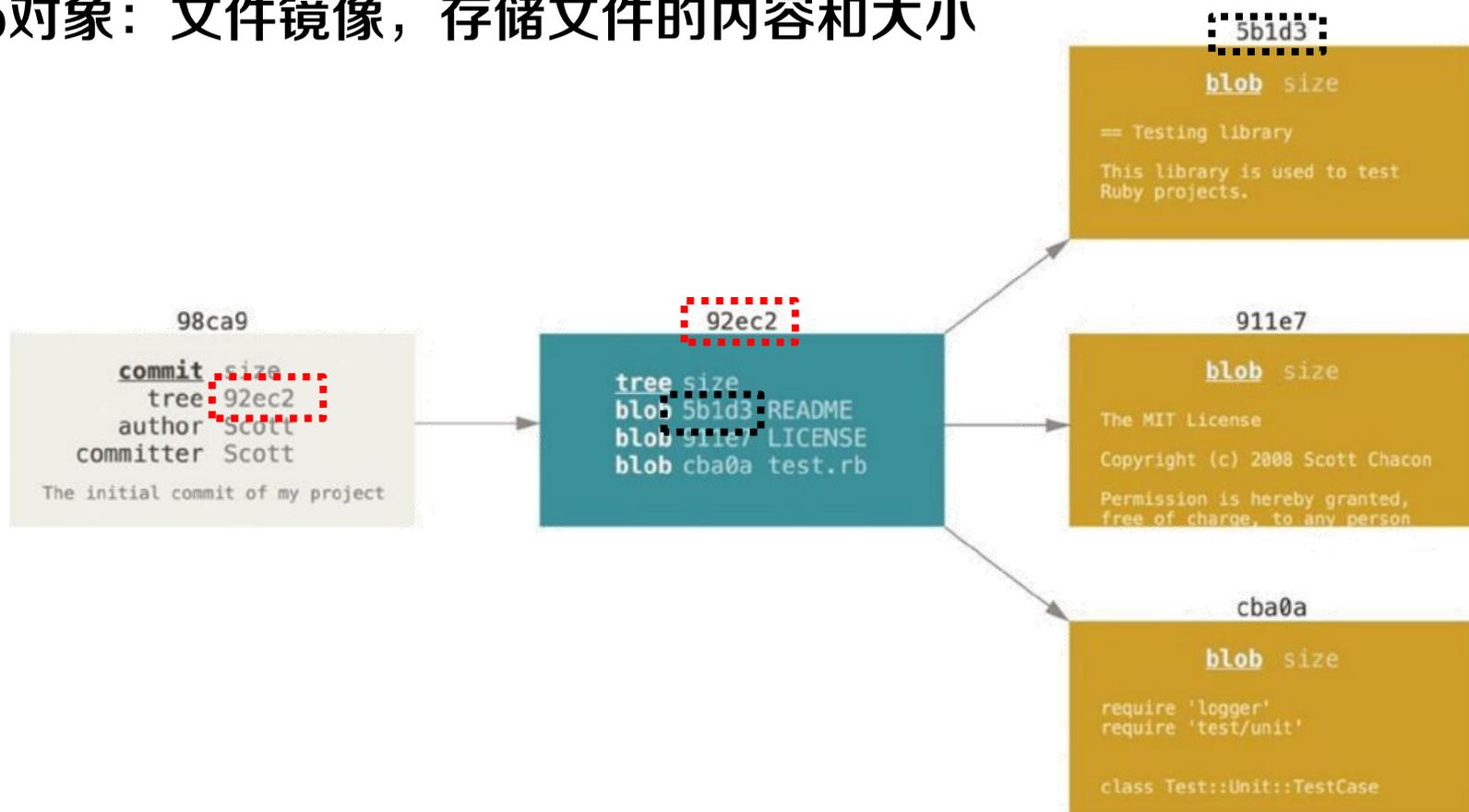
Merge branch 'HYB/peach' into 'HYB/peach' ...
由HYB编写于3星期前 ed5db8a8

09 1月, 2023 10次提交

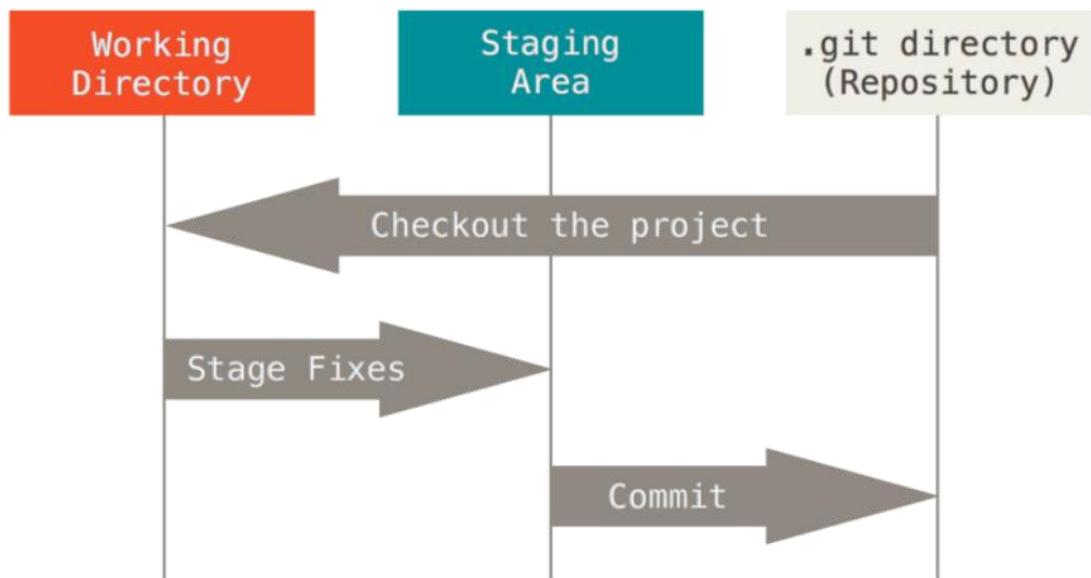
2
由fengtong xing编写于3星期前 e55e5bff

1
由fengtong xing编写于3星期前 3c3b0699

- Git内部的数据结构——基于SHA-1的Object database
 - Commit对象：保存本次提交版本的SHA-1校验和，拥有指向Tree对象的指针
 - Tree对象：记录着目录结构和 Blob 对象索引，拥有指向Blob和Tree对象的指针
 - Blob对象：文件镜像，存储文件的内容和大小



- Git中的文件状态：已修改、已暂存、已提交
 - 已修改（modified）表示修改了文件，但**未保存**到本地数据库中
 - **已暂存（staged）**表示已对当前版本文件进行了**标记**，使之包含在下次提交的快照中
 - `.git/index`文件
 - 已提交（committed）表示文件安全地**保存**在本地数据库中



description	2022/11/15 17:02	文件	1 KB
FETCH_HEAD	2022/11/15 17:07	文件	1 KB
HEAD	2022/11/15 17:03	文件	1 KB
index	2022/11/20 15:58	文件	5,670 KB
ms-persist.xml	2023/2/1 10:39	XML 文档	1 KB
ORIG_HEAD	2022/11/15 17:07	文件	1 KB
packed-refs	2022/11/15 17:03	文件	1 KB

T	目标	多人协作开发
I	输入	1. Git 2. 远程Git仓库
P	处理	1. 克隆远程Git仓库 2. 编程、开发 3. 保存至暂存区 4. 提交至本地Git仓库 5. 推送至远程Git仓库
O	输出	1. 得到本地Git仓库 2. 更新远程Git仓库

北京林业大学

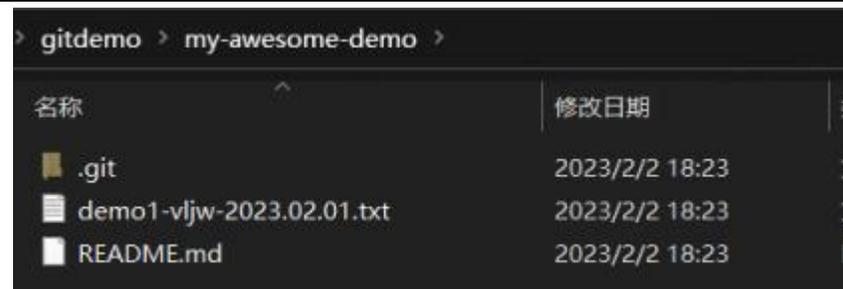
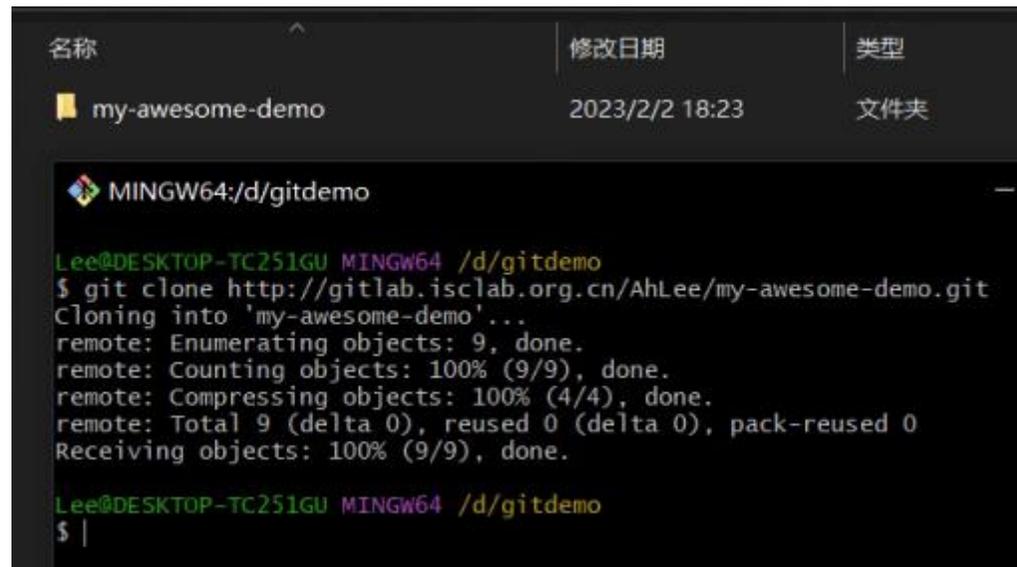
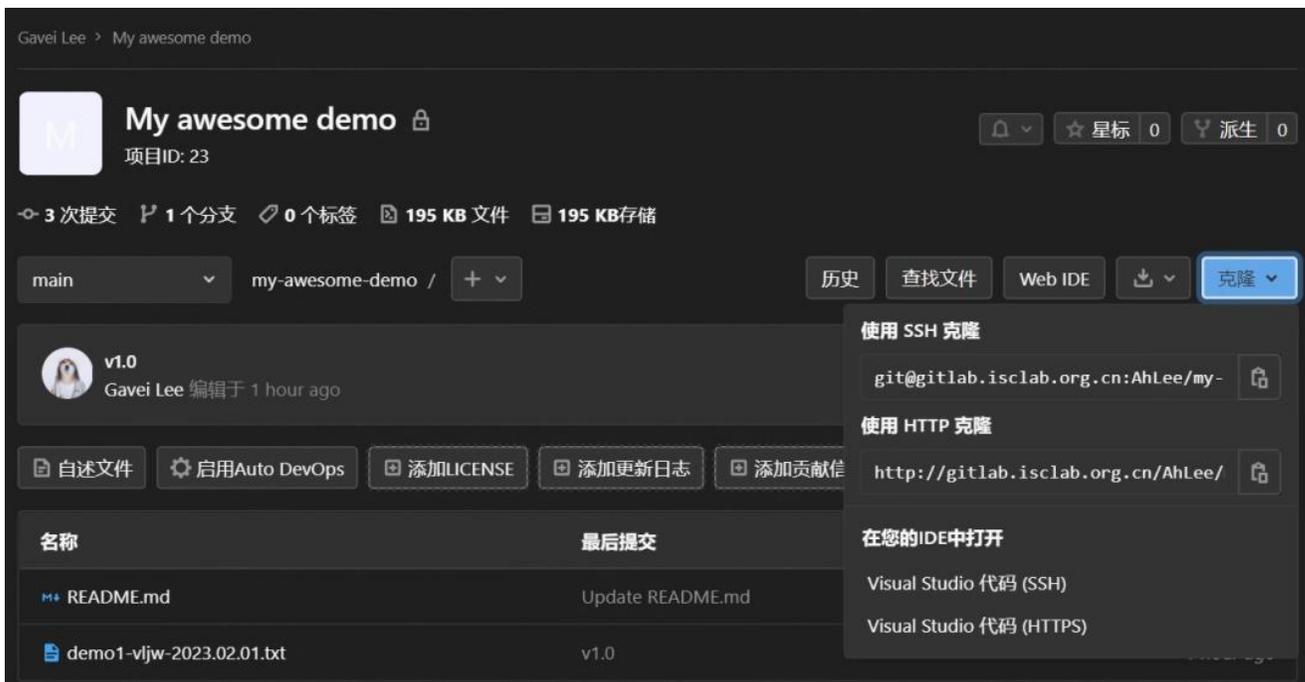


Git使用v1.0

- 获取Git仓库

- 克隆一个已经存在的Git仓库

- `git clone <url>` : 从远端仓库下拉某个Git仓库的**所有数据**，然后从中读取最新版本的文件的拷贝放到工作区



- 查看当前Git仓库状态

- git status

- 已跟踪和未跟踪

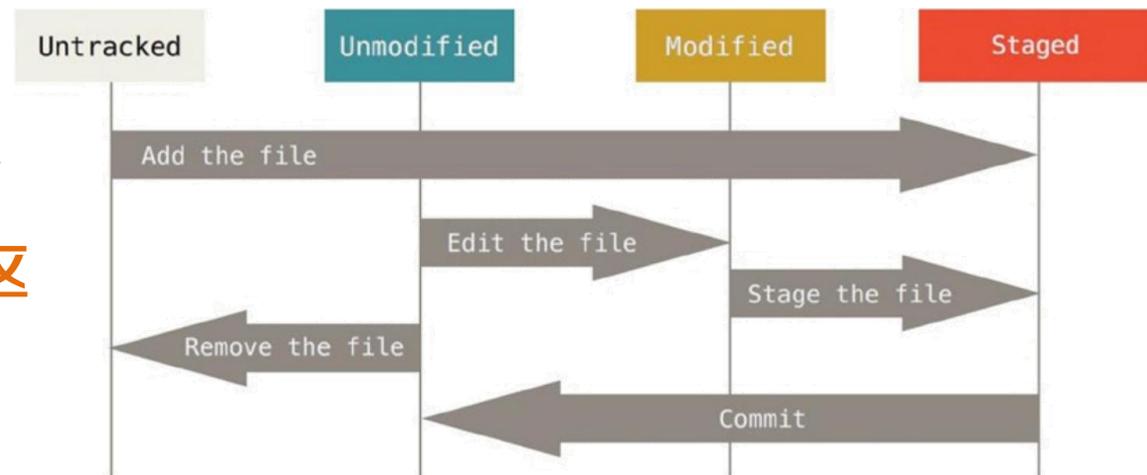


- 跟踪新文件或者把已跟踪的文件放到**暂存区**

- git add <file>

- 不再追踪某个文件

- git rm <file>



名称	修改日期	类型	大小
.git	2023/2/2 18:31	文件夹	
demo1-vljw-2023.02.01.txt	2023/2/2 18:23	文本文档	1 KB
demo2-vljw-2023.02.02.txt	2023/2/2 18:30	文本文档	0 KB
README.md	2023/2/2 18:23	MD 文件	1 KB

```
MINGW64:/d/gitdemo/my-awesome-demo
Lee@DESKTOP-TC251GU MINGW64 /d/gitdemo/my-awesome-demo (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
  demo2-vljw-2023.02.02.txt

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

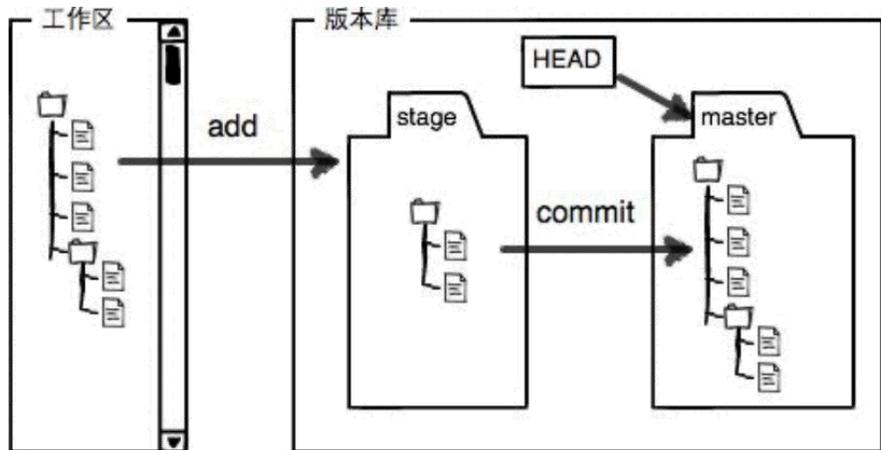
```
Lee@DESKTOP-TC251GU MINGW64 /d/gitdemo/my-awesome-demo (main)
$ git add demo2-vljw-2023.02.02.txt

Lee@DESKTOP-TC251GU MINGW64 /d/gitdemo/my-awesome-demo (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   demo2-vljw-2023.02.02.txt

Lee@DESKTOP-TC251GU MINGW64 /d/gitdemo/my-awesome-demo (main)
$
```

- 提交至本地Git仓库
 - `git commit -m <message>` :
提交暂存区快照到**本地Git仓库**



```
Lee@DESKTOP-TC251GU MINGW64 /d/gitdemo/my-awesome-demo (main)
$ git add demo2-vljw-2023.02.02.txt

Lee@DESKTOP-TC251GU MINGW64 /d/gitdemo/my-awesome-demo (main)
$ git commit -m "新增txt文件: demo2"
[main 3c00280] 新增txt文件: demo2
1 file changed, 1 insertion(+)
create mode 100644 demo2-vljw-2023.02.02.txt
```

- 将本地Git仓库的内容推送至**远程Git仓库**
 - `git push <remote> <branch>`

```
Lee@DESKTOP-TC251GU MINGW64 /d/gitdemo/my-awesome-demo (main)
$ git push origin main
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 355 bytes | 355.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To http://gitlab.isclab.org.cn/AhLee/my-awesome-demo.git
c40640a..3c00280 main -> main

Lee@DESKTOP-TC251GU MINGW64 /d/gitdemo/my-awesome-demo (main)
$
```

新增txt文件: demo2
Gavei Lee 编辑于 16 minutes ago

自述文件 启用Auto DevOps 添加LICENSE 添加更新日志 添

名称	最后提交
README.md	Update README.md
demo1-vljw-2023.02.01.txt	v1.0
demo2-vljw-2023.02.02.txt	新增txt文件: demo2

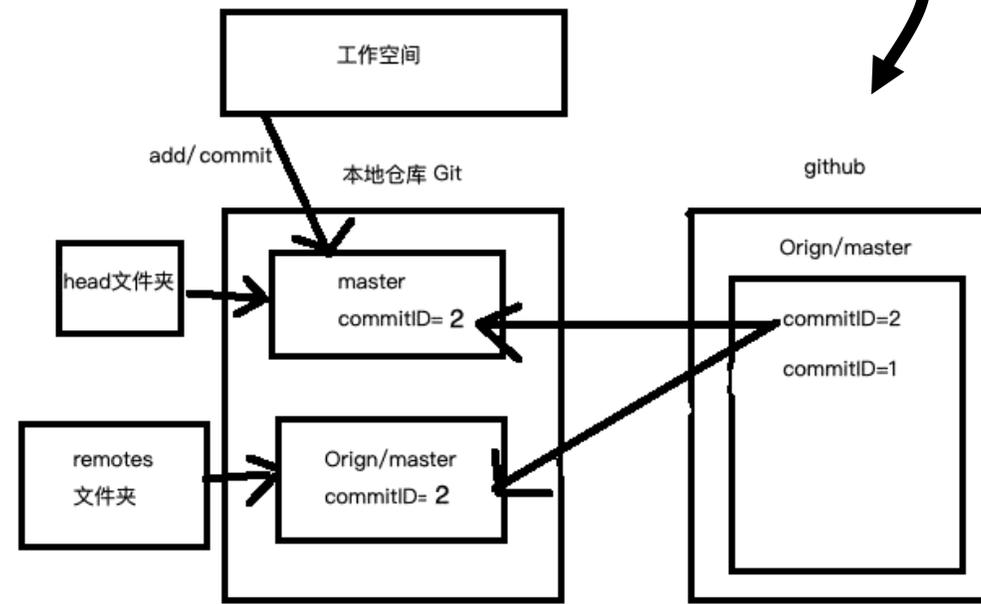
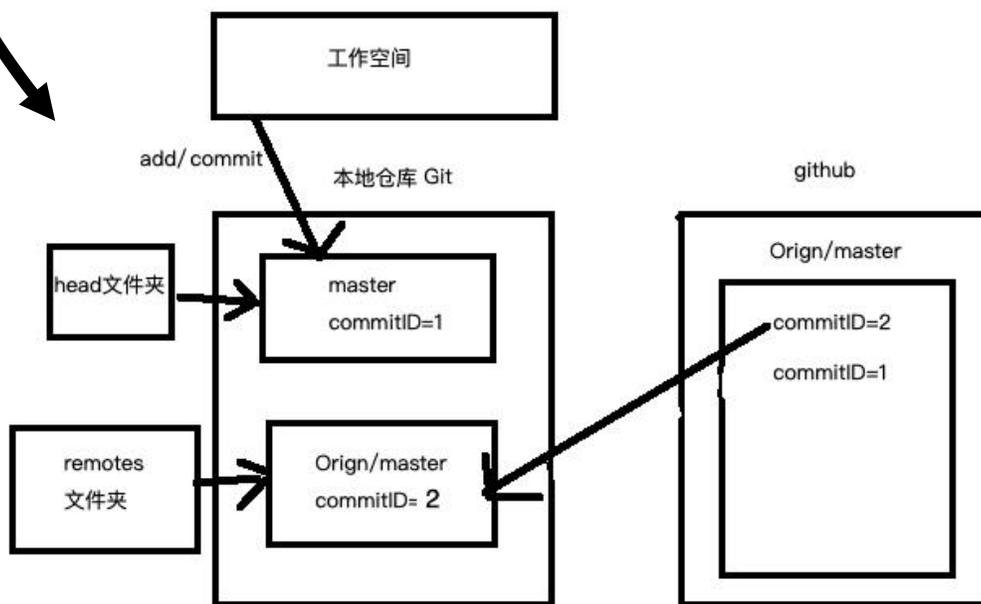
Git使用v2.0



Git使用v2.0

• 更新代码

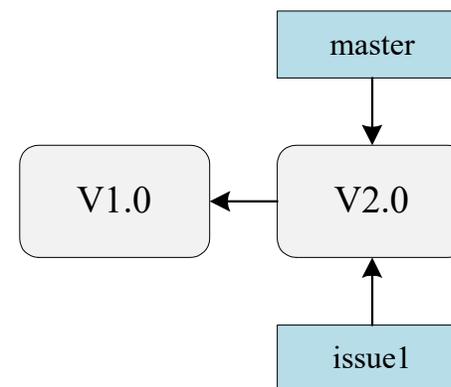
- git fetch: 将远程Git仓库里的所有内容同步到本地Git仓库
 - 不会自动将拉取后的文件与本机中任何文件合并，也不会修改正在进行的任何工作
 - 通常与git log和git merge配合使用（查看冲突日志，合并当前分支）
- git pull: 将远程Git仓库里的所有内容同步到本地Git仓库并**直接合并分支**



- 分支
 - 修改bug与开发新功能不冲突
 - 层次化管理
- Git分支
 - 本质上仅仅是指向Commit对象的**可变指针**
 - “Killer Feature”
 - Git默认分支名为master
- Git分支管理
 - 创建新分支
 - `git branch <branch-name>`
 - 本质上是创建了一个指向当前提交对象的指针



耐心渐渐消失



• Git分支管理

– HEAD指针

- 指向当前所在的本地分支

– 切换分支（切换HEAD指针）

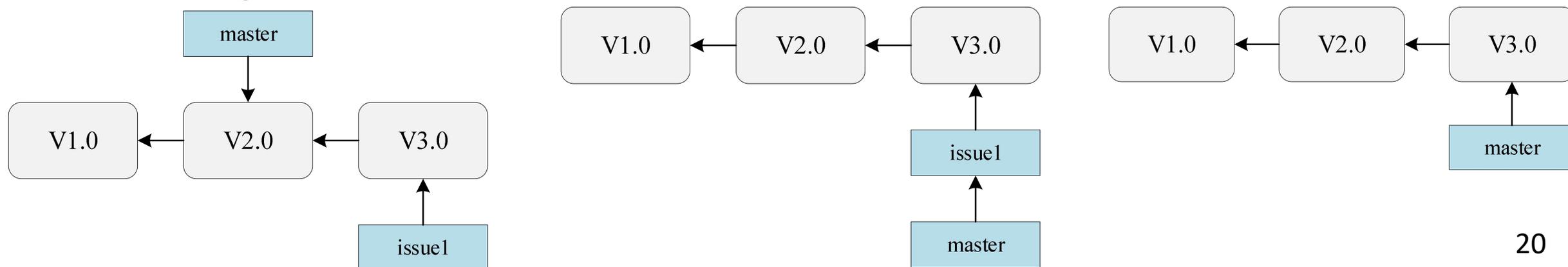
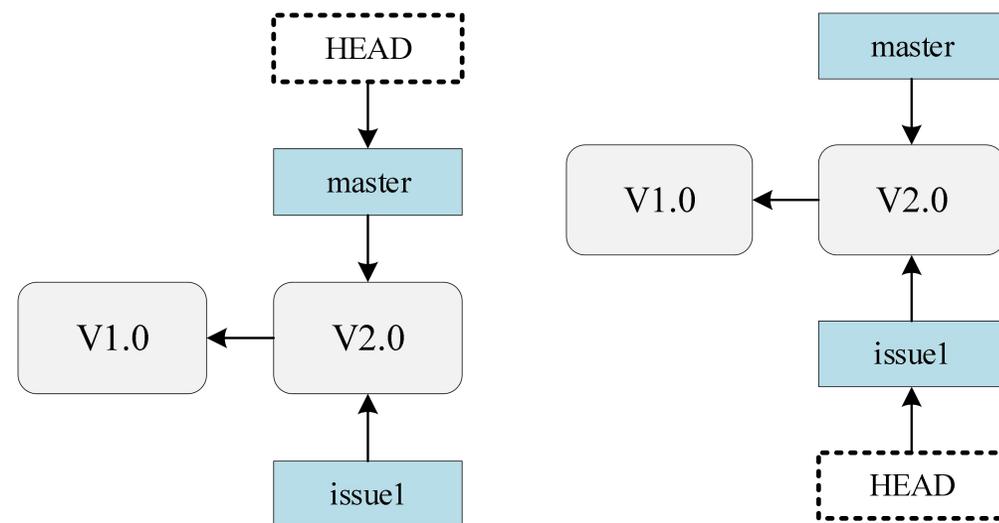
- `git checkout <branch-name>`

– 合并分支（Merging）

- `git merge <branch-name>`：将指定分支合并到**HEAD指向的分支上**

– 删除分支

- `git branch -d <branch-name>`



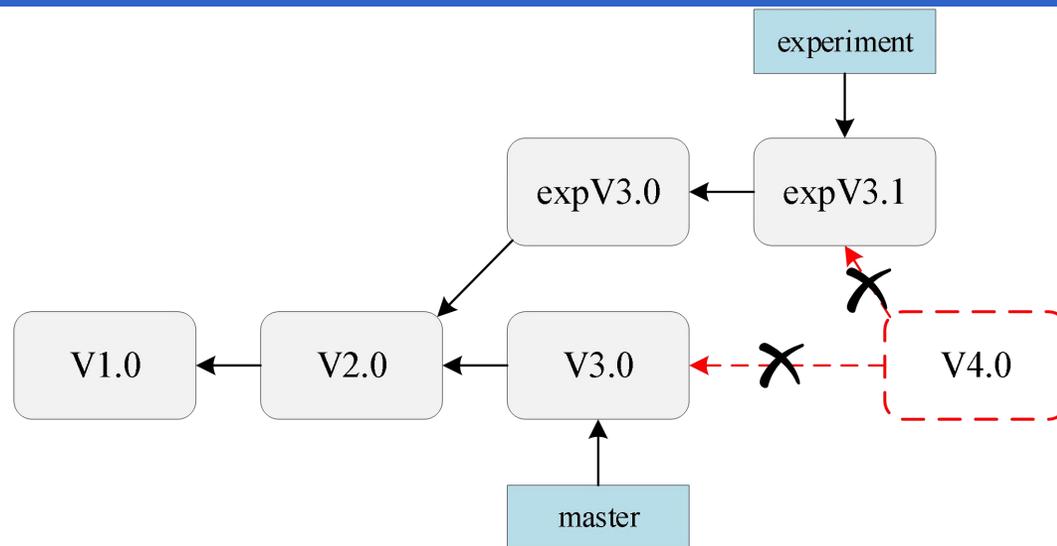
- 冲突

- 冲突问题

- three-way merge

- 解决冲突

- 基于Git的冲突**标记** (markers)
 - 冲突内容**手工合并**



```
demo2-vljw-2023.02.02.txt - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
1233321
10101010
00110011|

demo2-vljw-2023.02.02.txt - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
1245621
1010011
```

```
demo2-vljw-2023.02.02.txt - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
<<<<<<< HEAD
1233321
10101010
00110011
=====
1245621
1010011
>>>>>> experiment

MINGW64:/d/gitdemo/my-awesome-demo
Lee@DESKTOP-TC251GU MINGW64 /d/gitdemo/my-awesome-demo (main)
$ git merge experiment
Auto-merging demo2-vljw-2023.02.02.txt
CONFLICT (content): Merge conflict in demo2-vljw-2023.02.02.txt
Automatic merge failed; fix conflicts and then commit the result.

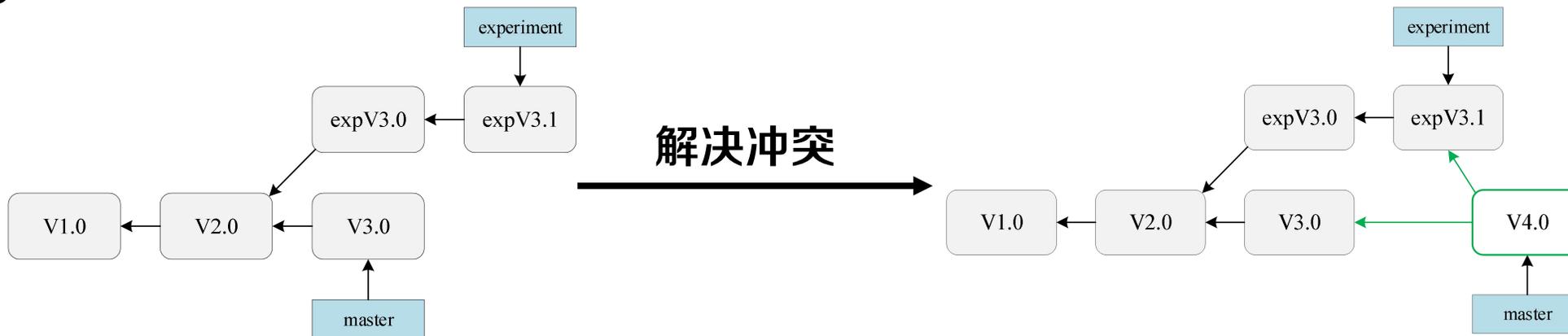
Lee@DESKTOP-TC251GU MINGW64 /d/gitdemo/my-awesome-demo (main|MERGING)
$ |

MINGW64:/d/gitdemo/my-awesome-demo
Lee@DESKTOP-TC251GU MINGW64 /d/gitdemo/my-awesome-demo (main)
$ git merge experiment
Already up to date.
```

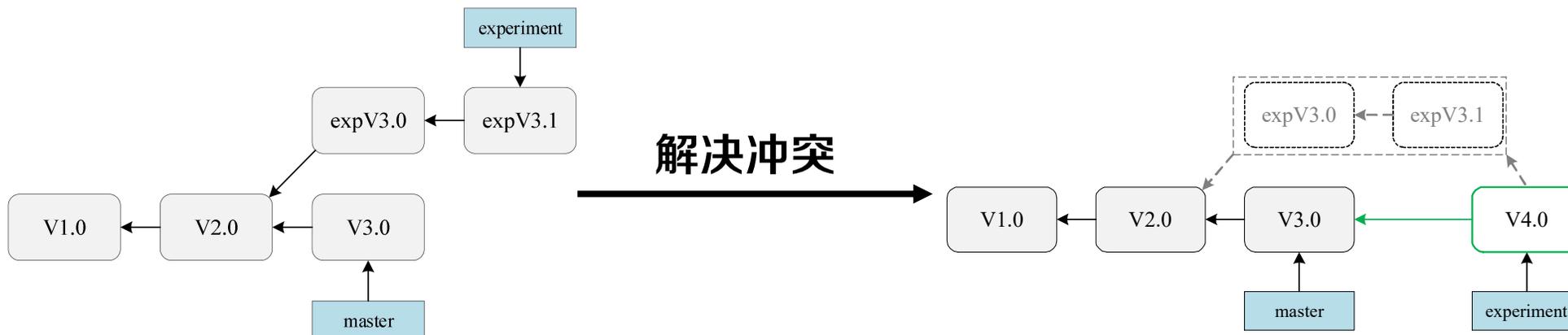
– Rebasing (变基)

- 将提交到某一分支上的所有修改都移至另一分支上
- `git rebase master`

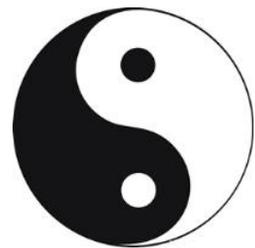
Merging



Rebasing



Git使用v3.0



Git使用v3.0

- 管理者视角

- 创建远程主仓库

- git init : 创建一个.git目录, 包含初始化的Git仓库中所有的必须文件, 但均未跟踪

- 制定开发规范

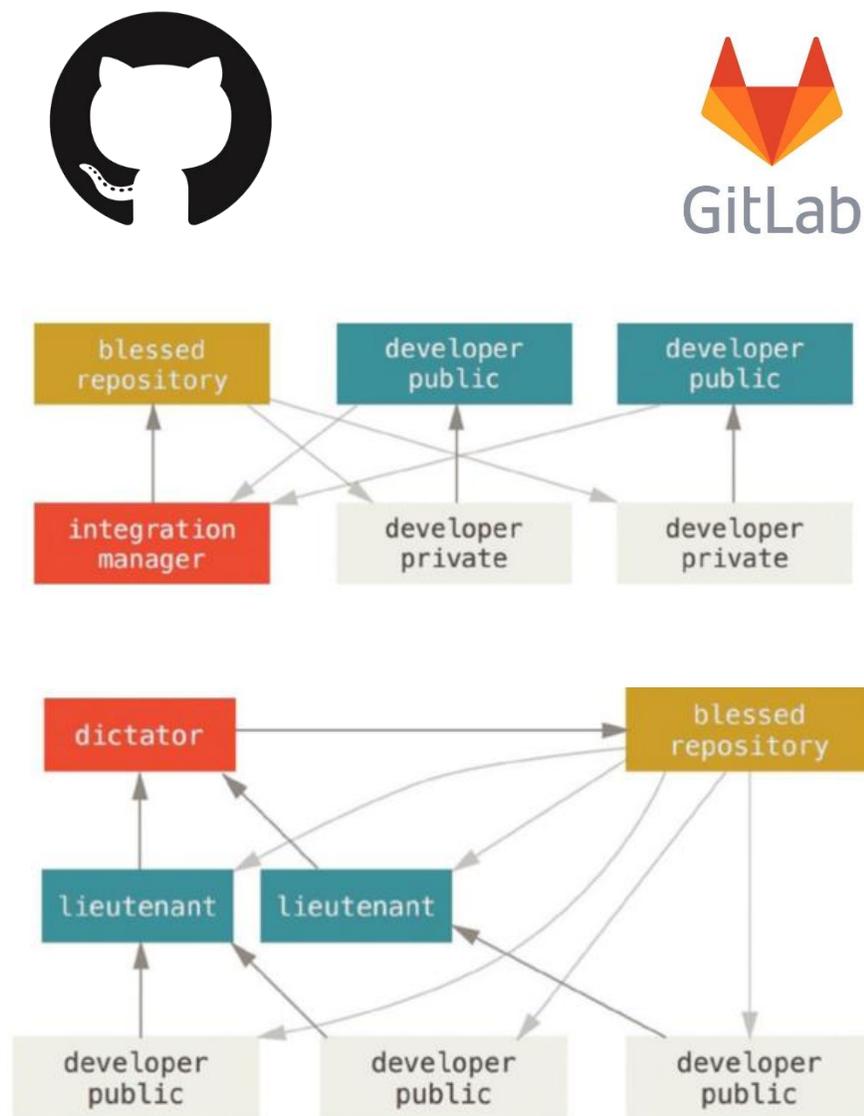
- 通知开发人员克隆主仓库并进行开发

- 接收开发人员的合并请求

- 维护本地主仓库

- 审核请求: 规范性、合理性
 - 解决冲突及合并分支
 - 维护主分支

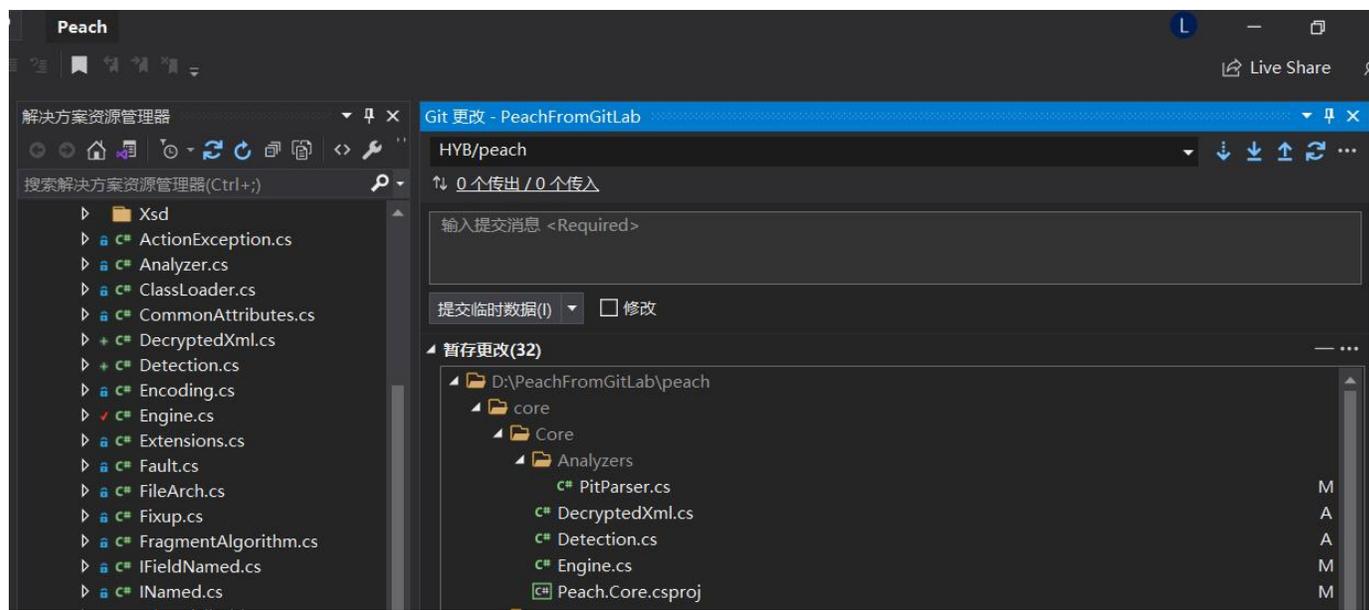
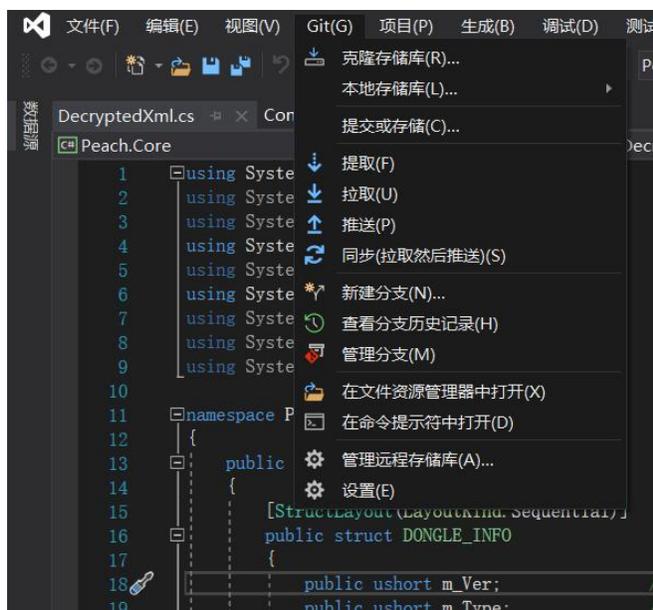
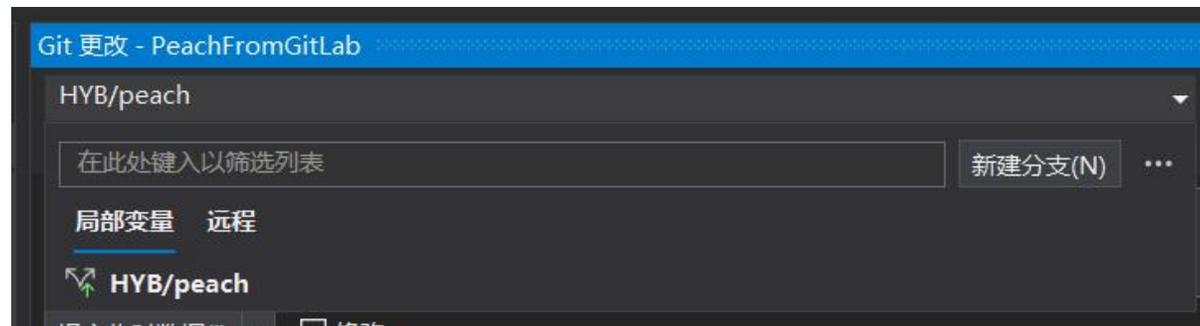
- 更新远程主仓库



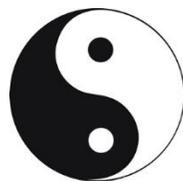
- Git中的hook操作
 - hook
 - 能在特定的重要动作发生之前或之后**触发**自定义脚本
 - Git Hook
 - 把命名正确且可执行的文件放入 `.git` 目录下的 `hooks` 子目录中
 - Git中默认提供的hook操作
 - `pre-commit` : 在输入`git commit`指令前被执行, 用于检查即将提交的快照等
 - `pre-receive` : 在接收来自客户端的**推送操作前**被执行, 用于校验日志格式等
 - `post-receive` : 在接收来自客户端的**推送操作后**被执行, 用于通知用户等
 -
 - Git中自定义hook操作
 - 添加命名相同的脚本文件
 - 直接修改`.sample`文件

名称	修改日期	类型
applypatch-msg.sample	2023/2/2 18:23	SAMPLE 文件
commit-msg.sample	2023/2/2 18:23	SAMPLE 文件
fsmonitor-watchman.sample	2023/2/2 18:23	SAMPLE 文件
post-update.sample	2023/2/2 18:23	SAMPLE 文件

- Git + Visual Studio
 - 可视化高
 - 无需Git指令
 - 操作简单



- Git + Pycharm



总结

- 总结

- Git使用v1.0

- 拉取至本地工作台
 - 提交至本地仓库
 - 推送至远程仓库

- Git使用v2.0

- 分支管理及合并
 - 冲突解决

- Git使用v3.0

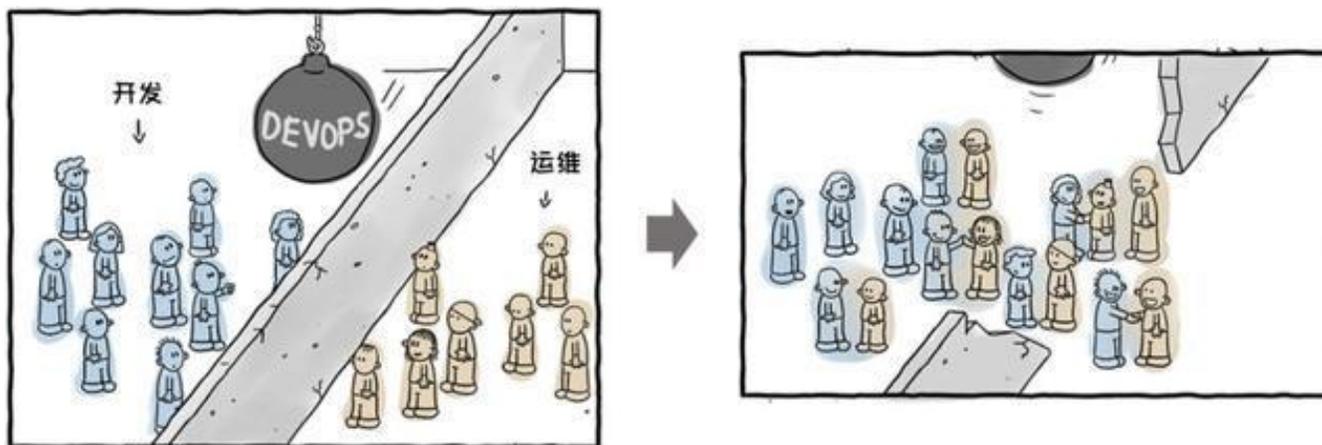
- 管理者视角
 - Git Hook操作
 - Git + VS/Pycharm

- 发展

- Git个性化及自动化

- Git + DevOps (**GitOps**)

- 核心思想：将应用系统的**声明性基础架构**和应用程序存放在Git版本库中



- [1] Chacon S, Straub B. Pro git[M]. Springer Nature, 2014.
- [2] Spinellis D. Git[J]. IEEE software, 2012, 29(3): 100–101.
- [3] 王真. 版本控制工具在软件开发项目管理中的应用——以 GIT 为例[J]. 项目管理技术, 2020, 18(06): 131–134.

谢谢!

大成若缺，其用不弊。大盈
若冲，其用不穷。大直若屈。
大巧若拙。大辩若讷。静胜
躁，寒胜热。清静为天下正。

