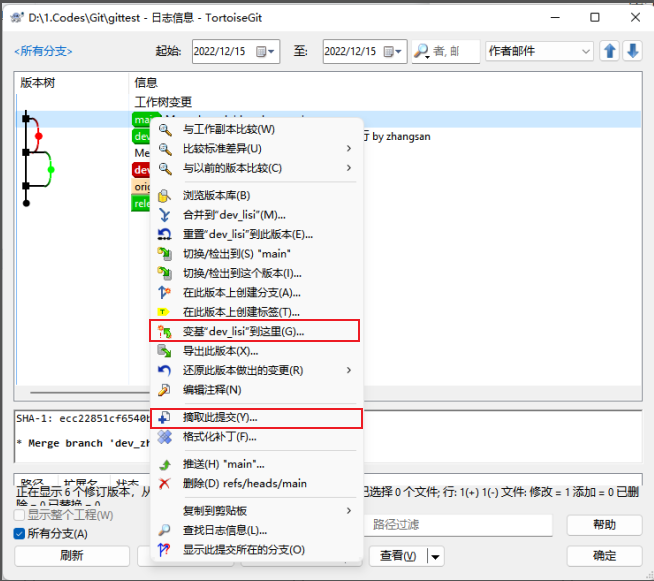
# Git变基和摘取

### 变基示例说明

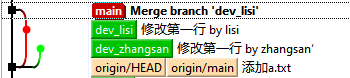
变基即git rebase，可视为将本地领先的提交“接续”到某个提交之后，下面以TortoiseGit可视化工具来示例一个变基操作，在TortoiseGit的日志窗口，可以很方便的查看分支图和进行变基操作，以及常用的摘取操作。



当一个分支很多人在使用时，不适合变基，如main、release；变基会破坏git树结构，远端分支变基后，所有跟踪此分支的其它仓库都需要变基（否则拉取下来会形成合并提交）。个人开发分支是适合变基的，因为不会影响他人且方便解决冲突。如下图，zhangsan和lisi都基于main分支修改了a.txt的第一行



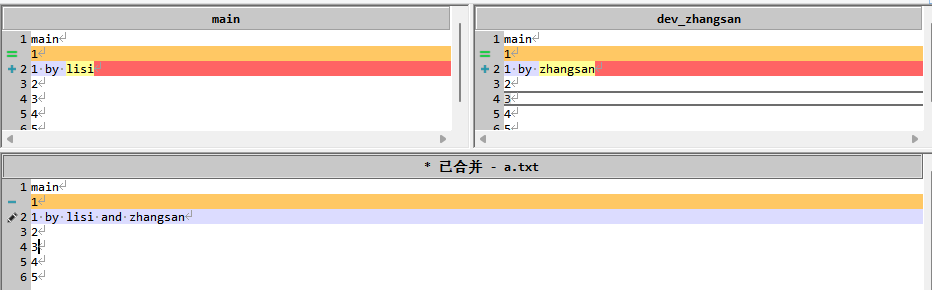
合并人员先处理了lisi的合并请求（按照GitLab默认做法以及通用做法，往main或者release分支的合并都是非fast-forward的（命令行参数：--no-ff），这样会形成一个合并提交，即下图中的“Merge branch 'dev\_lisi'”）



这时候如果处理zhangsan的合并请求，则会冲突，因为他们都修改了第一行，本地操作的提示如下：



由于合并人员都是在GitLab上操作，不方便查看和处理冲突，所以最好的办法是zhangsan自行处理冲突，将dev\_zhangsan上的提交“接续”（变基）到main后面，变基时，处理冲突，如下图：



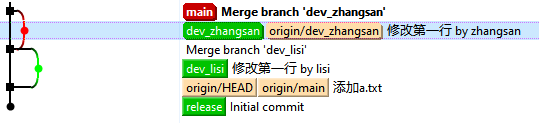
处理冲突后需要进行一次提交（注意，这次提交最好是只包含本次变基的内容以及处理冲突的内容，不要在此基础上再去添加其它开发内容，高效的使用Git是每次提交都是相对独立的内容），提交之后分支图如下，成功的将dev\_zhangsan“接续”到了main分支后面。



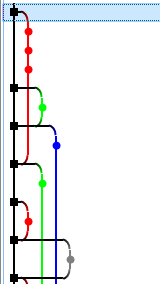
注意，如果dev\_zhangsan跟踪了远端分支origin/dev\_zhangsan，如上图红框部分，此时origin/dev\_zhangsan不是dev\_zhangsan的上游了，是无法直接推送dev\_zhangsan的。所以，**变基之后必须使用强制推送（git push -f origin dev\_zhangsan）覆盖远端origin/dev\_zhangsan**。



推送之后，创建合并请求，合并人员不需要解决冲突就可以合并了，合并后分支图如下：



至此，通过变基在本地解决冲突的流程完成，变基处理得当，仓库的分支图就会比较“好看”、容易理解，简单来说，就是永远保持合并方向为dev\_xxx-->main/release，尽量不要出现main/release-->dev\_xxx，如下图所示：

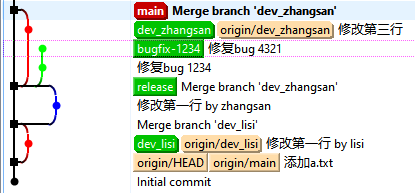


由于Git安装时，git pull的默认行为是git fetch + git merge，可以修改全局配置将git pull的默认行为修改为git fetch + git rebase，指令如下：

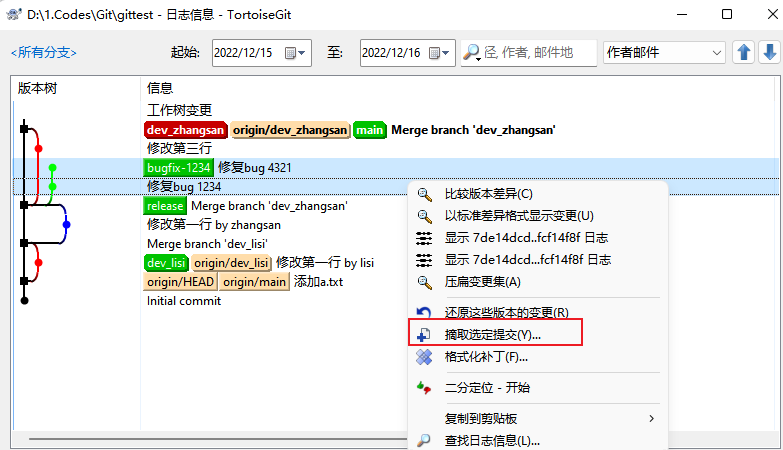
git config --global --add pull.rebase true

### 摘取示例说明

摘取即git cherry-pick，命令行摘取时需要带上提交的SHA，下面演示使用TortoiseGit在日志窗口进行摘取。如下图，bugfix-1234分支基于release分支修复了两个bug，进行了两次提交，为了在main分支修复同样的问题，需要将这两个提交摘取过来。



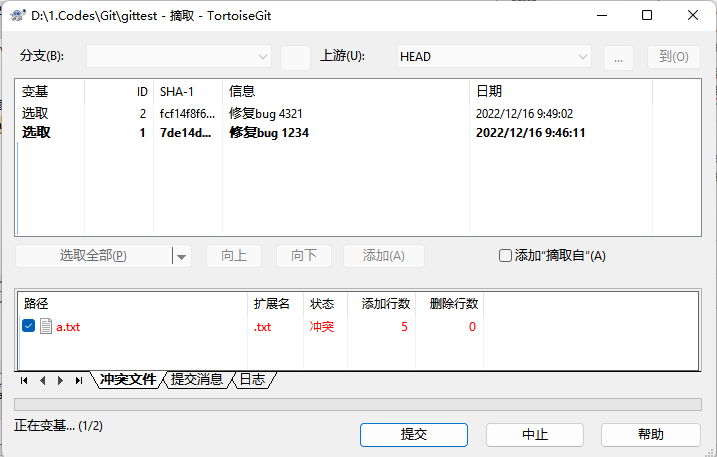
假定由zhangsan来摘取，首先把dev\_zhangsan变基到最新的main分支。然后摘取提交，同时选中两个提交，右键选择“摘取选定的提交”：



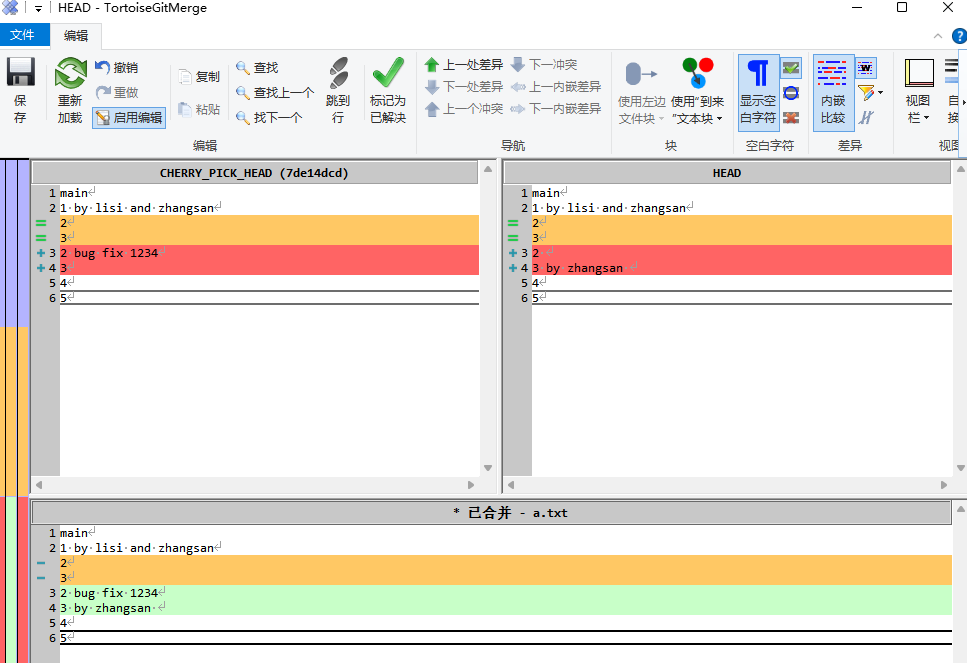
弹出摘取对话框，选择“继续”



如果有冲突则需要解决冲突，没有冲突则自动完成了摘取，下图是有冲突的情况：



双击a.txt，编辑冲突，解决之后点击工具栏“标记为已解决”：



回到原来的界面，选择“提交”，如果摘取的提交数量很多，则可能会出现多次冲突的情况，需要逐一解决并处理，最终在界面选择完成，完成之后按照流程提交合并请求即可。

### 总结

1. 注意使用git pull拉取最新代码时，默认的git pull = git fetch + git merge，git pull前应确保本地提交都已经推送，否则会造成一个本地合并，没有实际作用且导致分支图错乱。
2. 使用git最主要的是理解其版本管控的逻辑，也不需要很深入就能满足日常开发使用。
3. 可以在本地创建版本库，多多练习，也可以阅读ProGit等书籍。