

昵称: 灰太郎^_^

园龄: 5年2个月

粉丝: 25

关注: 0

[+加关注](#)

2020年12月						
日	一	二	三	四	五	六
29	30	1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31	1	2
3	4	5	6	7	8	9

搜索

 找找看

常用链接

[我的随笔](#)[我的评论](#)[我的参与](#)[最新评论](#)[我的标签](#)[更多链接](#)

我的标签

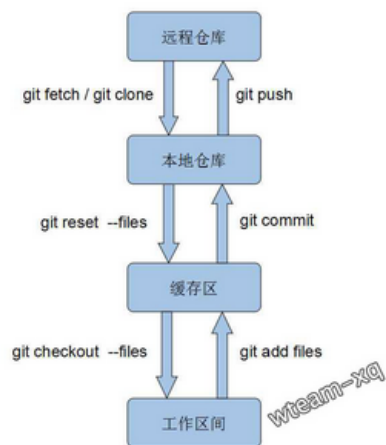
[Linux\(8\)](#)[Utils\(5\)](#)[多线程\(5\)](#)[sql优化\(4\)](#)[jms\(4\)](#)[messaging\(4\)](#)[JAVA基础\(4\)](#)[git\(3\)](#)[logback\(3\)](#)

git原理图解

目录

- 1.提交 代码到远程仓库
- 2.将远程仓库代码更新到本地
- 3.更新到本地仓库时， 出现冲突， 解决冲突
- 后记:

正文



本文背景，在实际项目中使用git已有一年，发现不少同事虽然会使用常用git指令，但并不理解每个指令对应的作用原理。今天静下心来总结下git 的基本理解：代码的存在区域；本文以实际项目出发，理清使用git过程中，代码的迁徙流程。

git跟传统的代码管理器（如:svn）不同，主要区别在于git多了个本地仓库以及缓存区，所以即使无法联网也一样能提交代码。术语解释：

工作区: 即我们创建的工程文件，在编辑器可直观显示；

Mac(3)
更多

随笔分类

logback(1)
ActiveMQ(4)
browser(1)
cookie&session(1)
Date(2)
Git(7)
interview(2)
JAVA(9)
Linux(8)
Mac环境(3)
Maven(1)
Mybatis(3)
MYSQL(7)
Nginx(2)
Redis(1)
更多

随笔档案

2018年11月(2)
2018年6月(1)
2018年1月(1)
2017年12月(4)
2017年11月(2)
2017年10月(2)
2017年8月(6)
2017年7月(3)
2017年6月(4)
2017年4月(2)
2017年3月(1)
2017年1月(2)
2016年12月(4)
2016年10月(1)
2016年8月(2)
更多

最新评论

1. Re:Lombok之使用详解
链式结构和build结构啥区别啊
--大东胖了

2. Re:logback 配置详解 (一) ——logger、root
有理有例，非常清晰，赞！！
--搬砖的老刘

3. Re:logback 配置详解 (一) ——logger、root
说的很清楚
--雨vs枫

缓存区: 只能通过git GUI或git shell 窗口显示，提交代码、解决冲突的中转站；

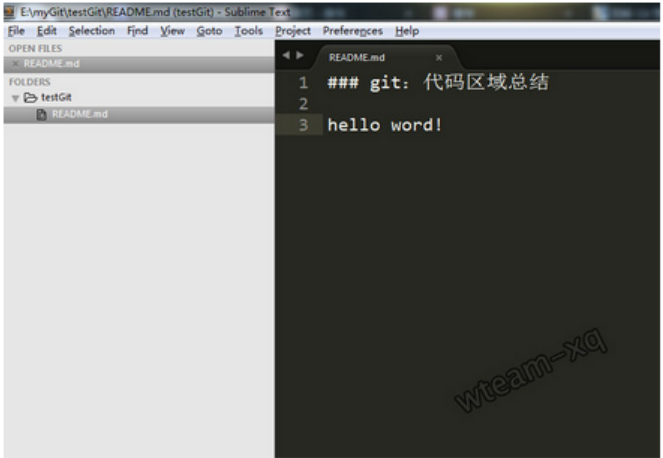
本地仓库: 只能在git shell 窗口显示，连接本地代码跟远程代码的枢纽，不能联网时本地代码可先提交至该处；

远程仓库: 即保存我们代码的服务器，本文以公共版本控制系统：**github**为例，登录github账号后可直观显示；

接下来，我们以三个实际操作例子讲解git的日常，代码如何在上述4个区域流动。

1.提交 代码到远程仓库

首先在本地区工作区间创建一个新工程：testGit，然后在项目里新建一个README.md，工作区间的工程如下：



远程仓库创建一新工程，具体操作参见 [如何在window上把你的项目提交到github](#)



将该新建的工程提交至远程仓库，关键的git 指令如下：

[回到顶部](#)

4. Re:logback 配置详解 (一) ——logger、root
谢谢博主, 这篇文章让我明白了root logger appender三者之间的关系。

--科学民主自由

5. Re:git原理图解

将本地修改放入缓存区(成功后本地工作区间的代码跟本地仓库代码会同步), 具体指令:

git stash

上述原文这里用git add可以吗

--给点阳光y

阅读排行榜

1. logback 配置详解 (一) ——logger、root(49514)
2. git原理图解(22599)
3. 使用Sonatype Nexus搭建Maven私服后如何添加第三方JAR包? (16963)
4. Mybatis Generator (定制化) 代码生成器(13877)
5. 多线程的安全问题(13512)

评论排行榜

1. logback 配置详解 (一) ——logger、root(5)
2. Lombok之使用详解(2)
3. 多线程的安全问题(2)
4. Mybatis Generator (定制化) 代码生成器(1)
5. git原理图解(1)

推荐排行榜

1. logback 配置详解 (一) ——logger、root(17)
2. xml语法、DTD约束xml、Schema约束xml、DOM解析xml(3)
3. git原理图解(2)
4. Spring框架 [一] ——spring概念和ioc入门 (ioc操作xml配置文件) (1)
5. 在使用Java8并行流时的问题分析(1)

```
git init
git add README.md
git commit -m "首次提交代码"
git remote add origin https://github.com/wteam-xq/testGit.git
git push -u origin master
```

指令解释:

`git init` 表示在当前的项目目录中生成本地的git管理;

`git add README.md` 将"README.md"文件保存至**缓存区**, 实际开发中一般使用`git add -A`, 使用-A:将新增、删除、修改的文件改动全保存至缓存区;

`git commit -m "first commit"` 将代码从**缓存区**保存至**本地仓库**, 实际开发中一般使用`git commit -am "说明的文字"`, 使用 -a: 如果没文件更改操作(增、删、改名) 就可以省略git add指令;

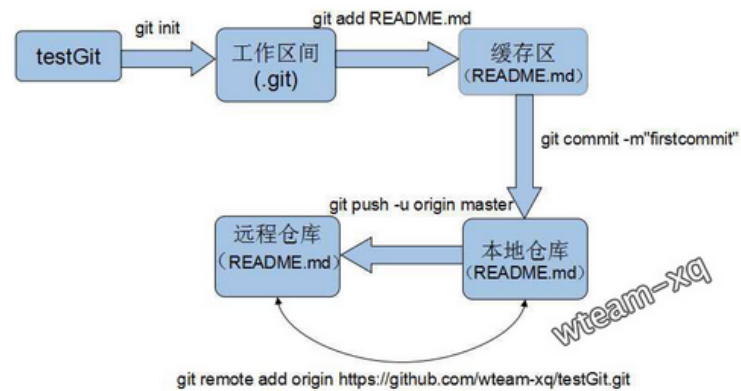
`git remote add origin https://github.com/wteam-xq/testGit.git`将**本地仓库**与指定的**远程仓库**创建 联系;

`push -u origin master` 将**本地仓库**代码推送至**远程仓库**, 实际开发中 该指令后需要输入github 账号以及密码。(首次提交注意别遗漏`-u`指定默认主机)

以上指令正常执行后, 本地仓库的代码就提交到远程仓库了:



原理图如下:



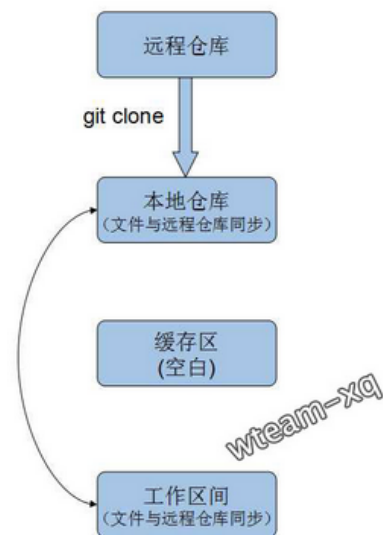
[回到顶部](#)

2.将远程仓库代码更新到本地

首先我们新建一文件夹：copyTestGit，进入该文件夹后使用git 指令：

```
git clone https://github.com/wteam-xq/testGit
```

指令执行完毕后，就在该文件夹下生成一份副本啦（相当于多人协作时另一台设备上的工程文件），原理图如下：



接下来, 讨论`git pull`、`git fetch`、`git merge`的关系

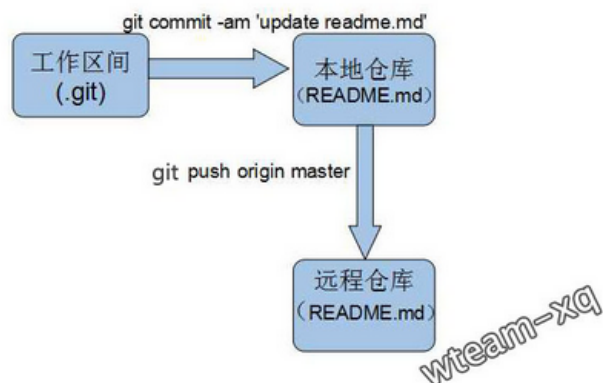
先抛简单结论:

```
git pull
#等同于下面命令
git fetch
git merge
```

实际项目: 我们在testGit工程中修改README.md,然后更新、提交下代码 执行以下git 指令(日常使用中会用`git status`看看是否有文件需要`git add`):

```
git commit -am 'update readme.md'
git push origin master
```

原理图如下:



远程仓库代码更新后, 我们进入另一本地仓库: copyTestGit\testGit, 将远程仓库的代码更新至该本地仓库。

在该目录下输入以下git指令:

```
git fetch
git merge origin/master
```

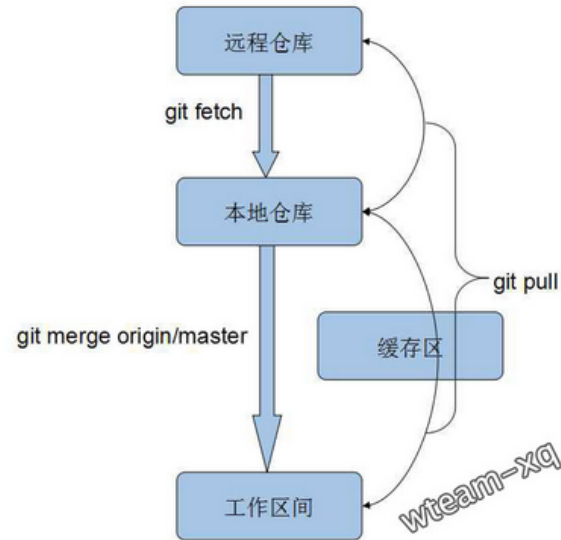
日常使用图方便一般都是直接:

```
git pull
```

以上指令的详细探讨请看 [少用pull,多用fetch 和 merge](#)

(注意: 本文注重git工作原理图不考虑多分支情况, 且使用了git clone所以副本工程已经跟主分支建立了追踪关系, 所以'pull' 'fetch' 后都不接分支代码)

以上指令区别的原理图:



[回到顶部](#)

3.更新到本地仓库时, 出现冲突, 解决冲突

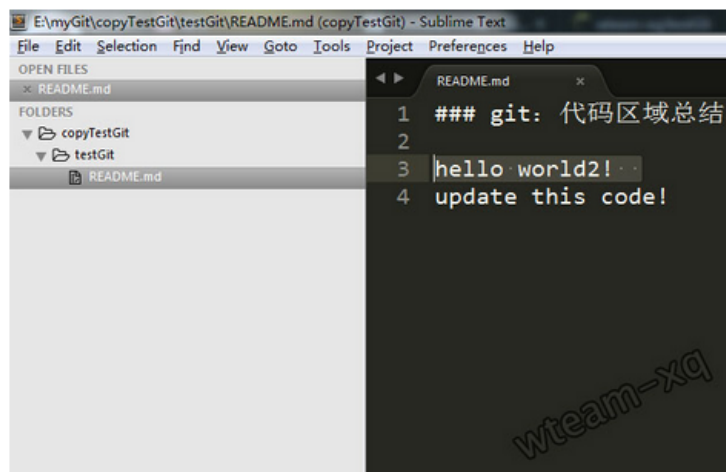
首先, 我们先重现下出现冲突的情况; 在testGit目录下先修改README.md文件第三行, hello word 修正为 hello world:

The screenshot shows a Sublime Text editor window titled 'E:\myGit\testGit\README.md (testGit) - Sublime Text'. The editor displays the contents of the README.md file, which are: 1. '### git: 代码区域总结', 2. an empty line, 3. 'hello world!', and 4. 'update this code!'. The third line is highlighted. The left sidebar shows the project structure with 'testGit' as the selected folder and 'README.md' as the selected file. A watermark 'wteam-xq' is visible in the bottom right corner of the editor window.

提交该修改到远程仓库（提交细节参照前述步骤）：



然后，在副本工程copyTestGit/testGit 目录下也修改README.md文件第三行，hello word 修正为 hello world2：



现在副本工程修改完了代码打算提交，提交前先远程仓库最新代码更新至本地仓库，惯例使用指令：

```
git pull
```

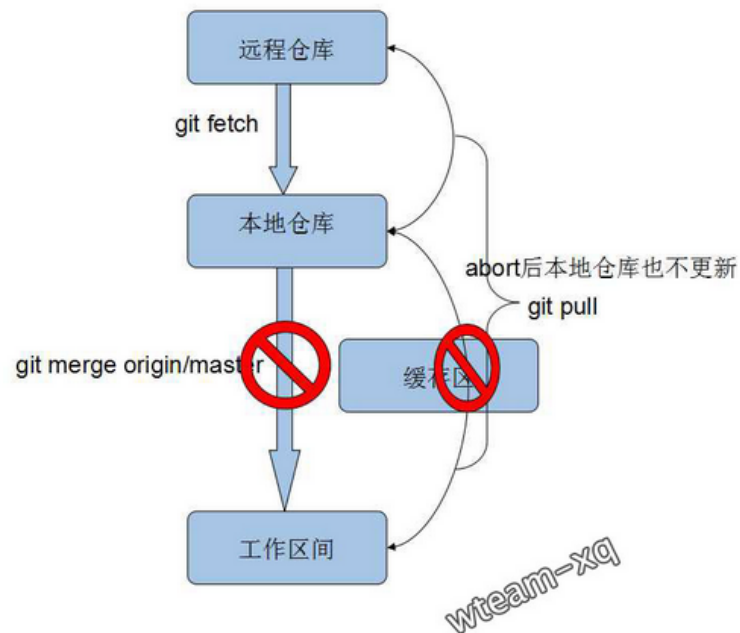
指令执行之后会发现以下冲突提示：




```
xq@XQ-PC /e/myGit/copyTestGit/testGit (master)
$ git pull
remote: Counting objects: 3, done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 3 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.
From https://github.com/wteam-xq/testGit
   2121b1f..dc6aaf0  master    -> origin/master
Updating 2121b1f..dc6aaf0
error: Your local changes to the following files would be overwritten by merge:
       README.md
Please, commit your changes or stash them before you can merge.
Aborting

xq@XQ-PC /e/myGit/copyTestGit/testGit (master)
$
```

出现以上提示，说明本次更新代码失败；主要在于本地工作区间跟远程仓库的新代码冲突了，图解如下：



接下来，有两种方式处理冲突：放弃本地修改或 解决冲突后提交本地修改

3.1 放弃本地修改

放弃本地修改意味着将远程仓库的代码完全覆盖本地仓库以及本地工作区间，如果对git的指令不熟悉那大可以将本地工程完全删除，然后再重新拷贝一次（`git clone`）。

当然，git如此强大没必要用这么原始的方法，可以让**本地仓库**代码覆盖本地修改，然后更新**远程仓库**代码；

本地仓库代码完全覆盖本地**工作区间**，具体指令如下：

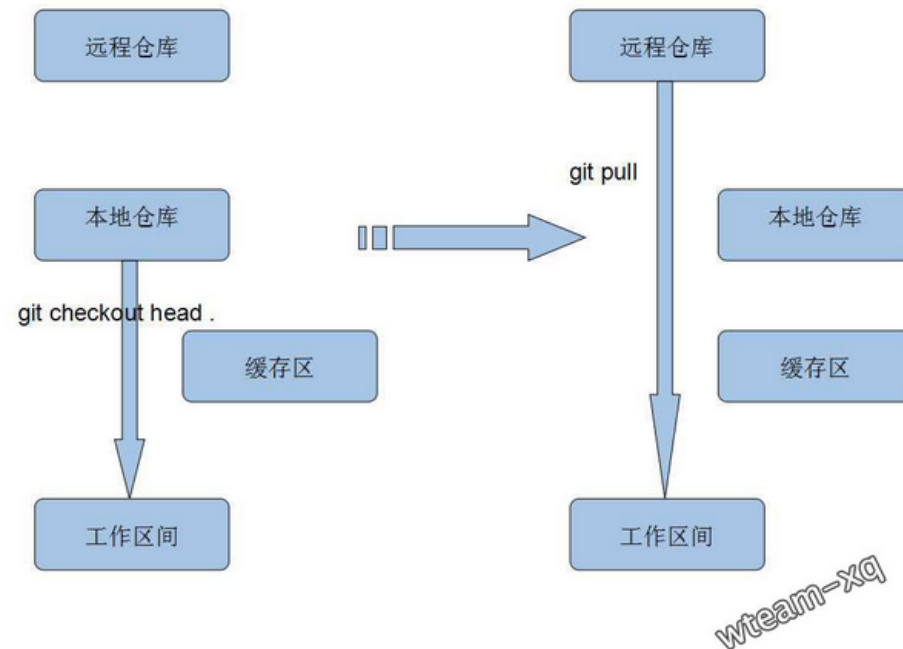
```
git checkout head .
```

(注意：别遗漏 "head" 后的 ".")

然后更新**远程仓库**的代码就不会出现冲突了：

```
git pull
```

原理图如下：



3.2 解决冲突后提交本地修改

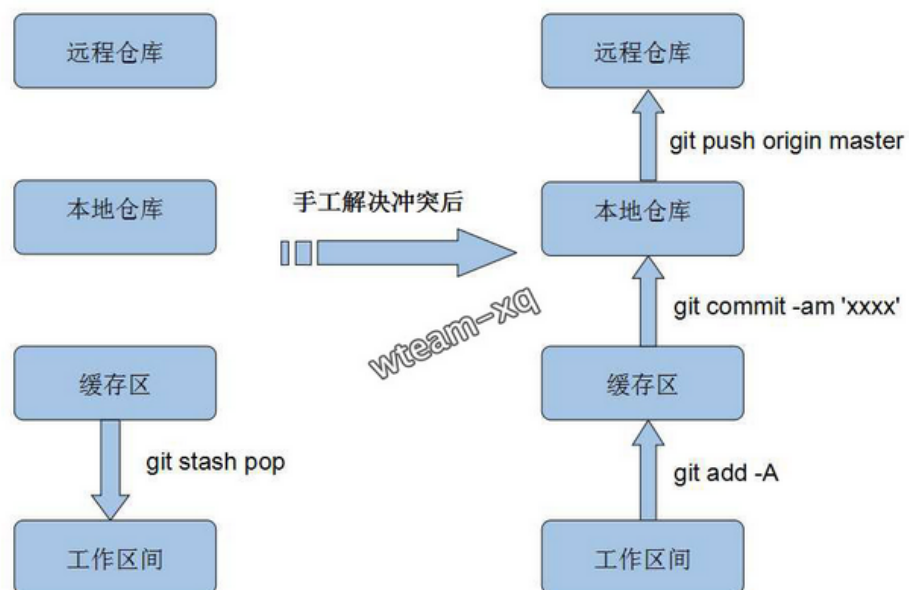
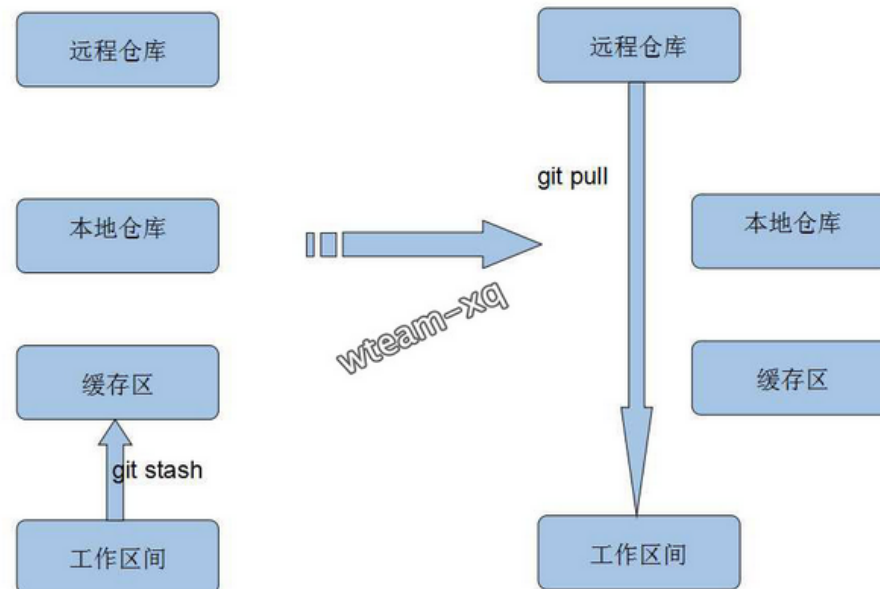
覆盖本地代码解决冲突方法适合不太懂git的菜鸟，像我这种git老鸟（其实并不是(¬_¬)）当然用更高级的git指令解决冲突。

细心的同学或许已发现，**缓存区**除了开始出现外，后续提交代码、更新代码篇章都在打酱油；终于，这次冲突解决事件，它将会是主角！

解决冲突后提交本地修改的思路大概如下：

将本地修改的代码放在**缓存区**，然后从**远程仓库**拉取最新代码，拉取成功后再从**缓存区**将修改的代码取出，这样最新代码跟本地修改的代码就会混杂在一起，手工解决冲突后，提交解决冲突后的代码。

原理图：



对应到我们实际项目中，进入 copyTestGit/testGit 执行指令`git pull`出现 (重回到上述冲突场景)

```
error: Your local changes to the following files would be overwritten by merge:
      README.md
Please, commit your changes or stash them before you can merge.
Aborting
```

将本地修改放入缓存区(成功后本地工作区间的代码跟本地仓库代码会同步)，具体指令：

```
git stash
```

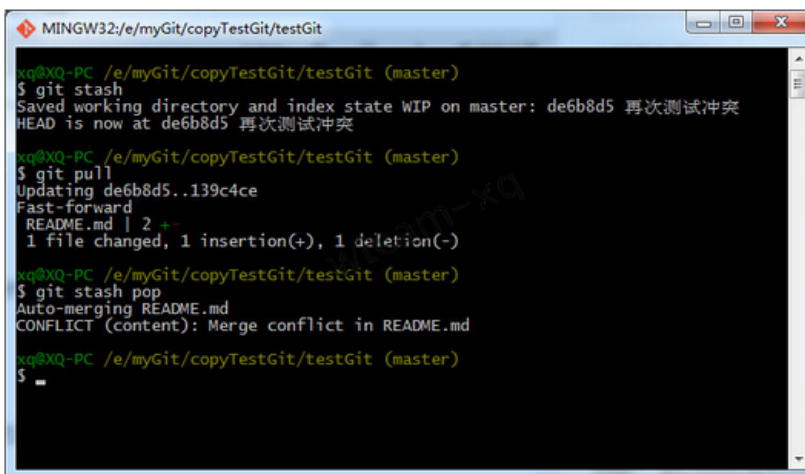
从远程仓库获取最新代码，具体指令：

```
git pull
```

然后，取出本地修改的代码，具体指令：

```
git stash pop
```

然后，git 自动合并冲突失败，冲突的代码就很清晰的展现在我们面前了：




```
MINGW32:/e/myGit/copyTestGit/testGit
xq@XQ-PC /e/myGit/copyTestGit/testGit (master)
$ git stash
Saved working directory and index state WIP on master: de6b8d5 再次测试冲突
HEAD is now at de6b8d5 再次测试冲突

xq@XQ-PC /e/myGit/copyTestGit/testGit (master)
$ git pull
Updating de6b8d5..139c4ce
Fast-forward
 README.md | 2 +-
 1 file changed, 1 insertion(+), 1 deletion(-)

xq@XQ-PC /e/myGit/copyTestGit/testGit (master)
$ git stash pop
Auto-merging README.md
CONFLICT (content): Merge conflict in README.md

xq@XQ-PC /e/myGit/copyTestGit/testGit (master)
$
```



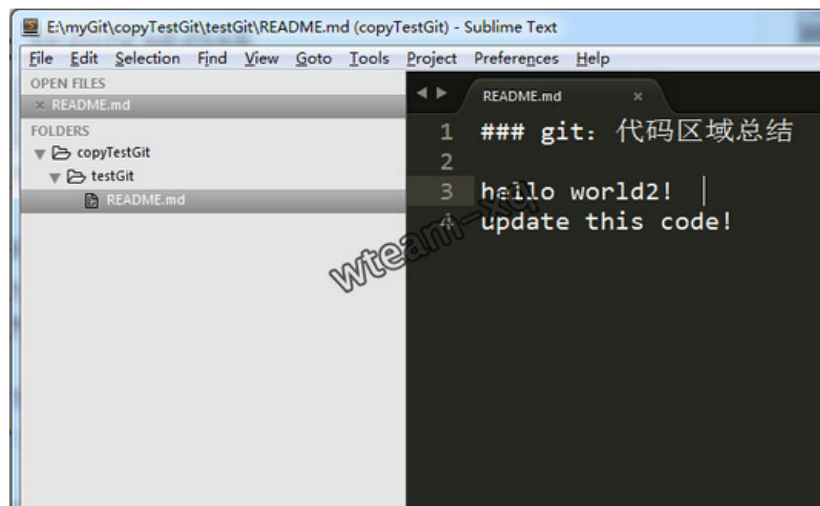
```
E:\myGit\copyTestGit\testGit\README.md (copyTestGit) - Sublime Text
File Edit Selection Find View Goto Tools Project Preferences Help
OPEN FILES
x README.md
FOLDERS
  copyTestGit
    testGit
      README.md

1 ### git: 代码区域总结
2
3 <<<<<<< Updated upstream
4 hello world!
5 =====
6 hello world2!
7 >>>>>>> Stashed changes
8 update this code!
```



(小广告：3.2栏更多细节请移步本人另一博文[git 代码冲突处理](#))

手工解决冲突：



告诉git，这个文件（README.md）的冲突 已经解决：

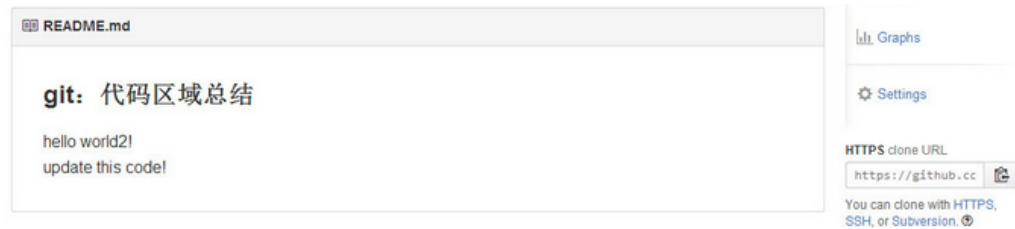
```
git add README.md
```

提交代码（细节参考前述流程）：

```
git commit -am '终于解决冲突啦！'
git push origin master
```

于是本地有冲突的代码就提交成功啦！

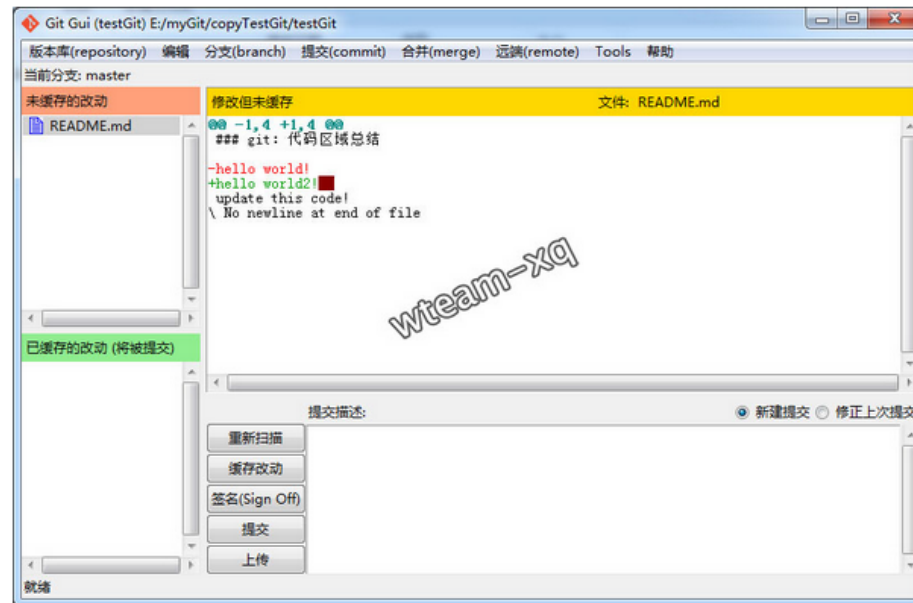




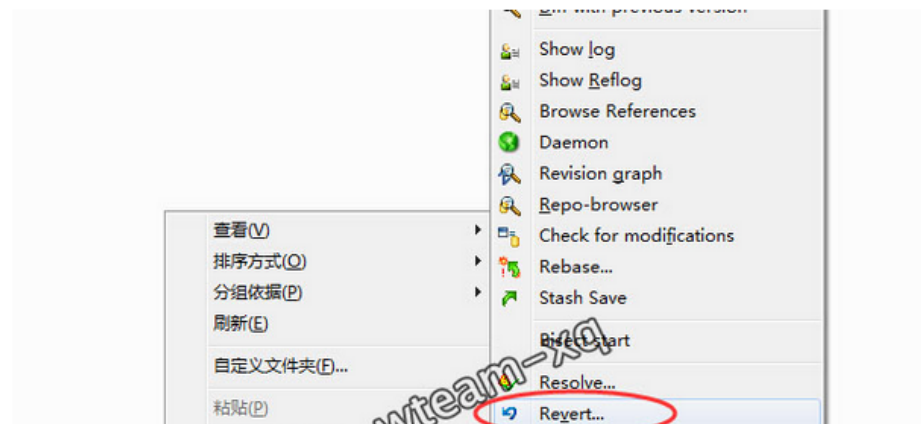
[回到顶部](#)

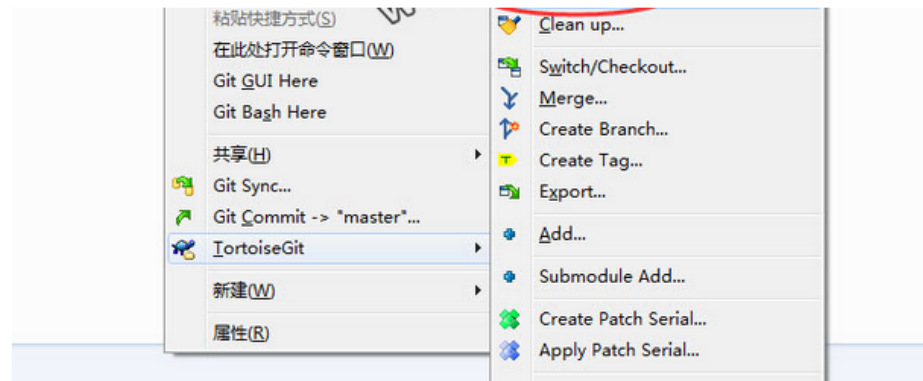
后记:

以上很多git指令适合在无图形化界面的linux中使用（例如：阿里云服务器操作git），实际开发中当然是用图形化界面解决！

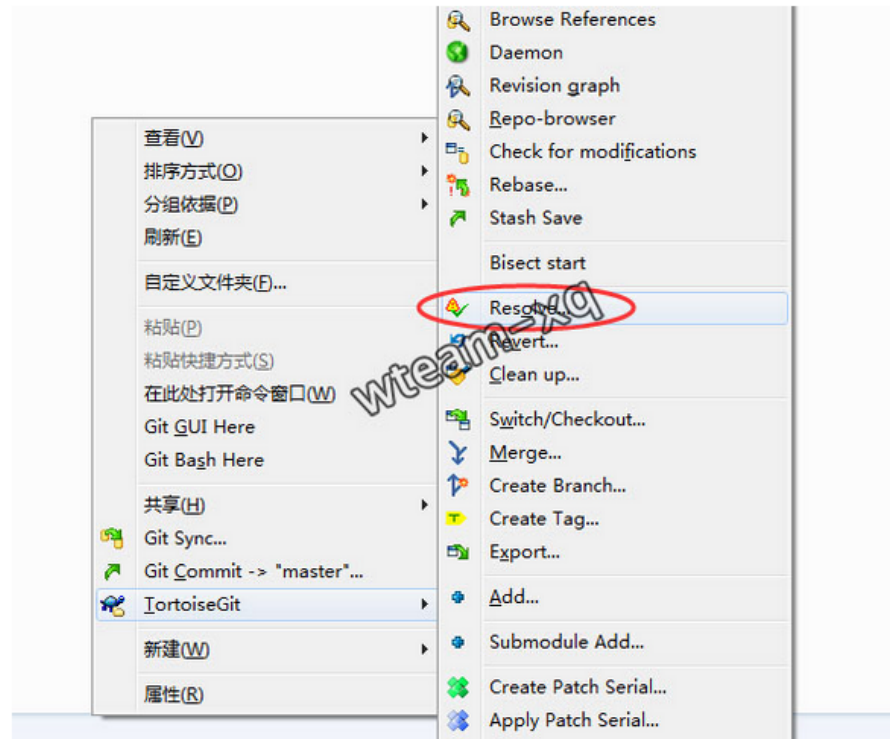


解决冲突之覆盖本地代码对应的是：工程目录下tortoiseGit(git 小乌龟)“Revert”：





解决冲突之解决冲突后提交本地修改对应的是：手工合并冲突代码后，工程目录下tortoiseGit(git 小乌龟)“resolve”：



参考文章：

1. [Git版本控制与 workflow](#)
2. [图解 Git 管理项目代码](#)
3. [如何在window上把你的项目提交到github](#)
4. [Git远程操作详解](#)
5. [Git 少用 Pull 多用 Fetch 和 Merge](#)

6.Git教程推荐一 (廖雪峰)

7.Git教程推荐二 (云溪)

8.Mac OS X Git安装教程

9.git-ssh 配置和使用

10.github设置添加SSH

11.生成多个git ssh密钥

12.ssh-agent 使用指南

分类: Git

好文要顶

关注我

收藏该文



灰太郎^_^

关注 - 0

粉丝 - 25

+加关注

2

推荐

0

反对

« 上一篇: git merge 和 git rebase 小结

» 下一篇: git fork

posted @ 2015-12-22 15:01 灰太郎^_^ 阅读(22600) 评论(1) 编辑 收藏

发表评论

#1楼 2020-07-04 09:39 | 给点阳光yh

回复 引用

“

将本地修改放入缓存区(成功后本地工作区间的代码跟本地仓库代码会同步), 具体指令:

git stash

上述原文这里用git add可以吗

支持(0) 反对(0)

刷新评论 刷新页面 返回顶部

发表评论

编辑 预览

B



支持 Markdown



提交评论

[退出](#) [订阅评论](#) [我的博客](#)

[Ctrl+Enter快捷键提交]

