**Git操作 - Git存储**

**1、什么是Git存储**

有时当你在项目的一个分支上，已经工作一段时间后，所有东西都进入了混乱的状态， 而这时你想要切换到另一个分支做一点别的事情。 问题是，你不想仅仅因为这点别的事，而把刚刚做了一半的工作进行一次提交。

针对这个问题，可以使用 git stash 命令来解决。（stash：贮藏）

git stash 命令会处理工作目录的脏状态，即：根据文件的修改与暂存的改动，然后将未完成的修改保存到一个栈上， 而你可以在任何时候重新应用这些改动（甚至在不同的分支上）。

**2、Git存储的常用命令**

1. git stash、git stash push、git stash save：保存当前工作进度，会把暂存区和工作区的改动stash（保存）起来。执行完这个命令后，在运行git status命令，就会发现当前工作目录中有一个干净的工作区，没有任何改动。
2. 使用git stash save '注释'可以给存储的进度添加注释。
3. git stash list：显示Git存储栈中的进度列表。
4. git stash pop [–index] [stash\_id]：Git存储栈会把工作区和暂存区的改动都恢复到工作区，通过git stash pop命令恢复进度后，同时会删除Git存储栈中当前进度的存储。
5. 1）git stash pop命令：如果不指定，Git会把最近一次的Git存储恢复到工作区。
6. 2）git stash pop --index命令：恢复最新的Git存储到工作区和暂存区，尝试将原来暂存区的改动，恢复到暂存区。
7. 3）git stash pop [stash\_id]命令：恢复指定的进度到工作区。stash\_id是通过git stash list命令得到的。例如：stash@{1}。
8. git stash apply [–index] [stash\_id]：不删除恢复的Git存储，其余和git stash pop命令一样。
9. git stash drop [stash\_id]：删除一个存储的进度。如果不指定stash\_id，则默认删除最新的存储进度。
10. git stash clear：删除所有存储的进度。
11. git stash show：显示stash的内容具体是什么，使用方法如 git stash show stash@{0}。
12. git stash branch <branchname> <stash>：基于进度创建分支。

|  |
| --- |
| 说明：  当我们执行git stash apply之后，发现所有的文件都变成了未暂存的，如果想维持原来的样子，也就是暂存过的依旧是暂存状态，那么可以使用 git stash apply --index命令。  即：--index选项作用，除了恢复工作区的文件外，还尝试恢复暂存区。  拓展：截至 2017 年 10 月下旬，Git 邮件列表上进行了广泛讨论，该讨论中弃用了 git stash save 命令， 代之以现有的 git stash push 命令。主因是 git stash push 引入了贮藏选定的 **路径规范** 的选项， 而有些东西 git stash save 不支持。  git stash save 不会很快就消失，所以不用担心它突然不见。 |

**3、常用参数说明**

1. -k：即--keep-index参数：
2. 在保存进度后，不会将暂存区重置。默认会将暂存区和工作区强制重置。
3. --patch：会显示工作区和HEAD的差异。通过对差异文件的编辑，决定在进度中，最终要保存的，工作区的内容。通过编辑差异文件，可以在进度中排除无关内容。
4. -u：即--include-untracked参数：
5. 默认情况下，git stash 只会存储已修改和暂存的 **已跟踪** 文件。 如果指定 --include-untracked 或 -u 选项，Git 也会存储任何未跟踪文件。
6. 然而在Git存储中，包含未跟踪的文件中，仍然不会包含明确 **忽略** 的文件。 要额外包含忽略的文件，请使用 --all 或 -a 选项。

**4、Git存储演示**

现在本地版本库中情况如下：

|  |
| --- |
| Bash L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/gitstash (dev) $ git log --oneline 4f69dbb (HEAD -> dev) 第3次提交，在dev分支新增stash.txt文件 89e03bd (master) 第2次提交，readme.txt文件，v2版本 80add2e 第1次提交，新增readme.txt文件  L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/gitstash (dev) $ git reflog 4f69dbb (HEAD -> dev) HEAD@{1}: commit: 第3次提交，在dev分支新增stash.txt文件 89e03bd (master) HEAD@{2}: checkout: moving from master to dev 89e03bd (master) HEAD@{3}: commit: 第2次提交，readme.txt文件，v2版本 80add2e HEAD@{4}: commit (initial): 第1次提交，新增readme.txt文件 |

有两分支，master分支和dev分支。

此时正在开发dev分支中的stash.txt文件。

下面开始演示 git stash 命令用法。

**1）从未完成工作dev分支，切换到master工作。**

|  |
| --- |
| Bash # 1.开发dev分支中的stash.txt文件。 L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/gitstash (dev) $ echo "stash.txt v2" >> stash.txt  # 2.查看工作目录中文件状态 L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/gitstash (dev) $ git status On branch dev Changes not staged for commit:  (use "git add <file>..." to update what will be committed)  (use "git restore <file>..." to discard changes in working directory)  modified: stash.txt  no changes added to commit (use "git add" and/or "git commit -a")  # 3.切换到master分支处理事情 L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/gitstash (dev) $ git checkout master error: Your local changes to the following files would be overwritten by checkout:  stash.txt Please commit your changes or stash them before you switch branches. Aborting  # 发现此时切换分支失败，让你在切换分支之前，先提交更改或存储更改。 # 下面我们进行存储更改  # 4.现在想要切换分支，但是还不想要提交之前的修改工作，所以存储修改。  # 将新的Git存储推送到栈上，运行 git stash 或 git stash push L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/gitstash (dev) $ git stash warning: LF will be replaced by CRLF in stash.txt. The file will have its original line endings in your working directory Saved working directory and index state WIP on dev: 4f69dbb 第3次提交，在dev分支新增stash.txt文件  # 5.查看Git存储栈中的内容，可以看到一条Git存储内容 L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/gitstash (dev) $ git stash list stash@{0}: WIP on dev: 4f69dbb 第3次提交，在dev分支新增stash.txt文件  # 6.再次查看工作目录中文件状态，可以看到工作目录是干净的，就可以进行分支切换了。 L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/gitstash (dev) $ git status On branch dev nothing to commit, working tree clean  # 7.此时我们先来查看一下历史提交记录的变化 # 没有变化 L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/gitstash (dev) $ git log --oneline  4f69dbb (HEAD -> dev) 第3次提交，在dev分支新增stash.txt文件 89e03bd (master) 第2次提交，readme.txt文件，v2版本 80add2e 第1次提交，新增readme.txt文件  # 此时提交历史多了一条HEAD移动记录，但是commit-id没有变化。（知道就行了） L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/gitstash (dev) $ git reflog 4f69dbb (HEAD -> dev) HEAD@{0}: reset: moving to HEAD 4f69dbb (HEAD -> dev) HEAD@{1}: commit: 第3次提交，在dev分支新增stash.txt文件 89e03bd (master) HEAD@{2}: checkout: moving from master to dev 89e03bd (master) HEAD@{3}: commit: 第2次提交，readme.txt文件，v2版本 80add2e HEAD@{4}: commit (initial): 第1次提交，新增readme.txt文件   # 7.切换到master分支。 L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/gitstash (dev) $ git checkout master Switched to branch 'master' |

**2）在master分支工作完成之后，在切换回dev分支。**

|  |
| --- |
| Bash # 1.在master分支，查看Git存储栈中的内容 L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/gitstash (master) $ git stash list stash@{0}: WIP on dev: 4f69dbb 第3次提交，在dev分支新增stash.txt文件  # 我们可以看到在master分支中，也能看到Git存储栈中的内容。 # 就说明Git存储栈中的内容可以在不同的分支使用。 # 这里就不在master分支使用了  # 2.切换到dev分支 L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/gitstash (master) $ git status On branch master nothing to commit, working tree clean  L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/gitstash (master) $ git checkout dev Switched to branch 'dev'  # 可以看到此时dev分支的工作目录是干净的。 L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/gitstash (dev) $ git status On branch dev nothing to commit, working tree clean  # 3.把Git存储栈中的内容进行恢复 L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/gitstash (dev) $ git stash list stash@{0}: WIP on dev: 4f69dbb 第3次提交，在dev分支新增stash.txt文件  L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/gitstash (dev) $ git stash apply stash@{0} On branch dev Changes not staged for commit:  (use "git add <file>..." to update what will be committed)  (use "git restore <file>..." to discard changes in working directory)  modified: stash.txt  no changes added to commit (use "git add" and/or "git commit -a")  # 4.查看stash.txt文件内容，可以看到内容恢复了 L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/gitstash (dev) $ cat stash.txt stash.txt v1 stash.txt v2 |

**3）删除Git存储栈中的内容**

|  |
| --- |
| Bash # 1.查看Git存储栈中的内容 L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/gitstash (dev) $ git stash list stash@{0}: WIP on dev: 4f69dbb 第3次提交，在dev分支新增stash.txt文件  # 我们可以看到Git存储栈中的内容，不会自动删除，是需要手动进行删除的。  # 2.删除Git存储栈中的内容 L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/gitstash (dev) $ git stash drop stash@{0} Dropped stash@{0} (00e9eb7b2ae3d1264656e75161b7c3f83b034282)  # 3.再次查看Git存储栈中的内容，可以看到没有任何存储的内容了。 L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/gitstash (dev) $ git stash list |

在工作中通常使用git stash pop命令，代替git stash apply和git stash drop命令。

|  |
| --- |
| 一般我们在实际工作中，不要把这个Git存储栈弄得很复杂，只建议存储一个元素，用完就删除。  参考：   * [https://git-scm.com/book/zh/v2/](https://links.jianshu.com/go?to=https%3A%2F%2Fgit-scm.com%2Fbook%2Fzh%2Fv2%2F) * [https://www.sunxiaoning.com/gongju/867.html](https://links.jianshu.com/go?to=https%3A%2F%2Fwww.sunxiaoning.com%2Fgongju%2F867.html) |

作者：繁华似锦Fighting 链接：https://www.jianshu.com/p/7e074bc8ad1e 来源：简书 著作权归作者所有。商业转载请联系作者获得授权，非商业转载请注明出处。