链滴

Git 基本操作

作者：shiweichn

原文链接：https://ld246.com/article/1533450265034

来源网站：链滴

Git基本操作

---

1. 删除文件

1.1 本地删除不是真正的删除

当前文件

```
[root@izwz9f5nsv33jsmjcx0qw1z demo]# ls
demo.txt  s.txt  welcome.log
[root@izwz9f5nsv33jsmjcx0qw1z demo]# rm *.txt
rm: remove regular file ‘demo.txt’? y
rm: remove regular empty file ‘s.txt’? y
[root@izwz9f5nsv33jsmjcx0qw1z demo]# ls
welcome.log
[root@izwz9f5nsv33jsmjcx0qw1z demo]# git ls-files
demo.txt
s.txt
welcome.log
[root@izwz9f5nsv33jsmjcx0qw1z demo]# git status
# On branch master
# Changes not staged for commit:
#   (use "git add/rm <file>..." to update what will be committed)
#   (use "git checkout -- <file>..." to discard changes in working directory)
#
#       deleted:    demo.txt
#       deleted:    s.txt
#
no changes added to commit (use "git add" and/or "git commit -a")
```

可以看出，直接删除工作区这些文件，但是这些文件在暂存区中仍然存在。从文件的状态来看，文件是在本地进行了删除，尚未添加到暂存区中。也就是说，在工作区删除文件对暂存区和版本库没有任影响。

若要恢复刚才在工作区删除的文件，用git checkout --<file> 。其实直接将工作区文件删除掉后，执行it add操作，再提交同样可以达到下面git rm 的效果。

1.2 执行 git rm 命令删除文件

```
[root@izwz9f5nsv33jsmjcx0qw1z demo]# ls
demo.txt  s.txt  welcome.log
[root@izwz9f5nsv33jsmjcx0qw1z demo]# git rm *.txt
rm 'demo.txt'
rm 's.txt'
[root@izwz9f5nsv33jsmjcx0qw1z demo]# git status
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
```

```
#
#       deleted:    demo.txt
#       deleted:    s.txt
#
[root@izwz9f5nsv33jsmjcx0qw1z demo]# git commit -am "delete txt file"
[master ecf4754] delete txt file
 2 files changed, 3 deletions(-)
 delete mode 100644 demo.txt
 delete mode 100644 s.txt
[root@izwz9f5nsv33jsmjcx0qw1z demo]# git ls-files --with-tree=HEAD^
demo.txt
s.txt
welcome.log
[root@izwz9f5nsv33jsmjcx0qw1z demo]# git cat-file -p HEAD^:demo.txt
123
qwe
中文测试
[root@izwz9f5nsv33jsmjcx0qw1z demo]#
```

通过git rm 命令删除文件就可以将删除动作加入暂存区，这时执行提交操作，就从真正意义上删除了件。

不过文件虽在版本库的最新提价中被删除，但是在历史提交中尚在。可以通过git ls-file --with-tree=EAD^ 命令查看历史版本中的文件列表，也可以通过git cat-file -p HEAD^:demo.txt 查看历史版本文件的内容。

---

2. 恢复删除的文件

虽然上面删除了文件并提交了。但是只是在最新的提交中删除了文件，我们还是可以从历史提交中恢文件。

```
[root@izwz9f5nsv33jsmjcx0qw1z demo]# ls
welcome.log
[root@izwz9f5nsv33jsmjcx0qw1z demo]# git cat-file -p HEAD~1:demo.txt > demo.txt
[root@izwz9f5nsv33jsmjcx0qw1z demo]# ls
demo.txt  welcome.log
[root@izwz9f5nsv33jsmjcx0qw1z demo]# cat demo.txt
123
qwe
中文测试
[root@izwz9f5nsv33jsmjcx0qw1z demo]# git status -s
?? demo.txt
[root@izwz9f5nsv33jsmjcx0qw1z demo]#
```

```
$ git show HEAD~1:demo.txt > demo.txt
$ git checkout HEAD~1 -- demo.txt
```

上面这俩命令也可以恢复文件。

---

3. 移动文件

```
[root@izwz9f5nsv33jsmjcx0qw1z demo]# ls
demo.txt  stest.txt  welcome.log
[root@izwz9f5nsv33jsmjcx0qw1z demo]# git status
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#       renamed:    s.txt -> stest.txt
#
[root@izwz9f5nsv33jsmjcx0qw1z demo]#
Display all 1472 possibilities? (y or n)
[root@izwz9f5nsv33jsmjcx0qw1z demo]# git commit -am "更名测试"
[master a8dc639] 更名测试
 1 file changed, 0 insertions(+), 0 deletions(-)
 rename s.txt => stest.txt (100%)
```

git mv命令可以完成改名操作，并且改名后查看状态有 renamed标记；在提交时可以看到前后两个件的相似度(百分比)。

其实git mv改名操作，相当于对旧文件执行删除(git rm)，对新文件执行添加(git add)如下：

```
[root@izwz9f5nsv33jsmjcx0qw1z demo]# mv demo.txt newdemo.txt
[root@izwz9f5nsv33jsmjcx0qw1z demo]# ls
newdemo.txt  s.txt  welcome.log
[root@izwz9f5nsv33jsmjcx0qw1z demo]# echo "rename file" >> newdemo.txt
[root@izwz9f5nsv33jsmjcx0qw1z demo]# git status
# On branch master
# Changes not staged for commit:
#   (use "git add/rm <file>..." to update what will be committed)
#   (use "git checkout -- <file>..." to discard changes in working directory)
#
#       deleted:    demo.txt
#
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#
#       newdemo.txt
no changes added to commit (use "git add" and/or "git commit -a")
[root@izwz9f5nsv33jsmjcx0qw1z demo]# git add -A
[root@izwz9f5nsv33jsmjcx0qw1z demo]# git status
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#       renamed:    demo.txt -> newdemo.txt
#
[root@izwz9f5nsv33jsmjcx0qw1z demo]# git commit -am "newdemo is from demo.txt"
[master 49dc15d] newdemo is from demo.txt
 1 file changed, 1 insertion(+)
 rename demo.txt => newdemo.txt (63%)
[root@izwz9f5nsv33jsmjcx0qw1z demo]#
```

git add -A 执行时，相当于对修改文件执行git add ,对删除文件执行git rm ， 对本地新增文件执行git add 。通过查看状态仍然可以看到renamed标记，并且提交时文件相似度变了，这是因为我中间改了文件内容。由此看出，Git的文件追踪能力真的很强。

---

4. 文件忽略

在工作区任意目录下创建.gitignore文件，将要忽略的文件写入改文件内。忽略只对未加入版本库的件有效。

使用--ignored 参数，才会在状态显示中看到被忽略的文件。git status --ignored -s。

4.1 本地独享式忽略文件

若.gitignore文件添加到版本库中后，且这个版本库共享给他人时， 这个忽略文件就会出现在他人的作区中，文件忽略在他人的工作区中同样生效。这样忽略文件就是共享式的。

独享式：

一种是针对版本库的 独享式 忽略，即在版本库 .git 目录下的一个文件，.git/info/exclude 来设文件忽略。

另一种是全局式的 独享式 忽略，即通过Git的配置变量core.excludesfile 指定一个忽略文件，其设置忽略对所有本地版本库均有效，如：git config --global core.excludesfile /home/sw/.gitignore

---

5. 文件归档

Git提供了一个创建归档的命令：git archive ,可以对任意提交对应的目录树创建归档。如：

基于最新的提交创建归档文件 latest.zip

git archive -o latest.zip HEAD

只将目录 src 和 doc 建立到归档文件中

git archive -o partial.tar HEAD src doc

在建立归档时，如果使用树对象ID进行归档，则使用当前时间作为归档中文件的修改时间，而如果使提交ID或里程碑等，则使用提交建立的时间作为归档中文件的修改时间。

如果使用tar 格式建立归档，并且使用提交ID或里程碑ID，还会把提交ID记录在归档文件的文件头中记录在文件头中的提交ID可以通过 git tar-commit-id命令获取。

如果在建立归档时想忽略某些文件或目录，可以通过为相应文件或目录建立export-ignore属性加以现。