

# 开源项目阅读与管理—版本管理

## 三、开源项目阅读与管理-版本管理

### 开源项目阅读与管理- 版本管理

步骤 1：工作区和暂存区

步骤 2：管理修改

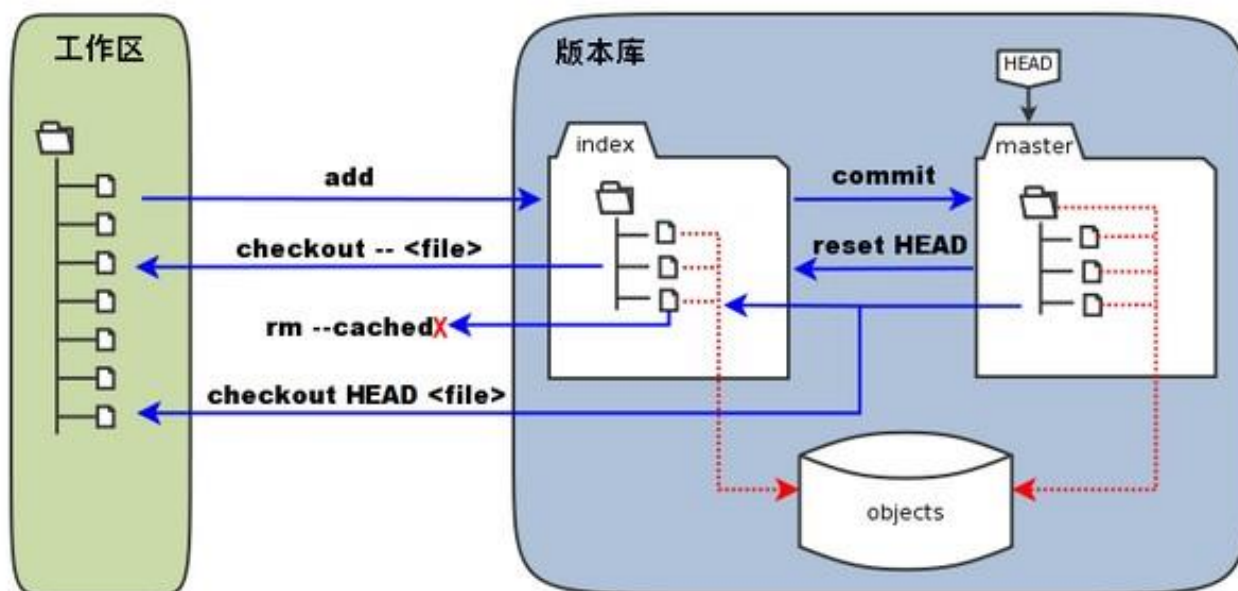
步骤 3：撤销修改

步骤 4：删除文件

### 步骤 1：工作区和暂存区

- **工作区**：就是你在电脑里能看到的目录。
- **暂存区**：英文叫 stage 或 index。一般存放在 `.git` 目录下的 `index` 文件（`.git/index`）中，所以我们把暂存区有时也叫作索引（index）。
- **版本库**：工作区有一个隐藏目录 `.git`，这个不算工作区，而是 Git 的版本库。

下面这个图展示了工作区、版本库中的暂存区和版本库之间的关系：



1、Git 和其他版本控制系统如 SVN 的一个不同之处就是有暂存区的概念。

先来看名词解释。

工作区（Working Directory）

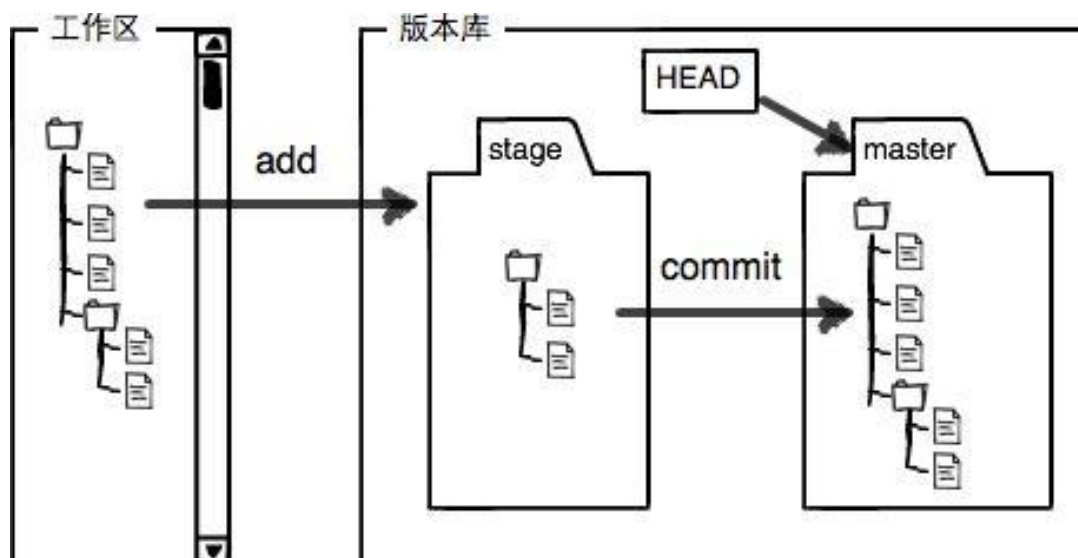
就是你在电脑里能看到的目录，比如我的 myrepo 文件夹就是一个工作区：

电脑 > 新加卷 (D:) > myrepo >	
名称	修改日期
.git	2022/6/
readme.docx	2022/6/

版本库（Repository）

工作区有一个隐藏目录.git，这个不算工作区，而是 Git 的版本库。

Git 的版本库里存了很多东西，其中最重要的就是称为 **stage**（或者叫 **index**）的暂存区，还有 Git 为我们自动创建的第一个分支 **master**，以及指向 **master** 的一个指针叫 **HEAD**。



前面讲了我们把文件往 Git 版本库里添加的时候，是分两步执行的：

第一步是用 **git add** 把文件添加进去，实际上就是把文件修改添加到暂存区；

第二步是用 **git commit** 提交更改，实际上就是把暂存区的所有内容提交到当前分支。

因为我们创建 Git 版本库时，Git 自动为我们创建了唯一一个 **master** 分支，所以，现在，**git commit** 就是往 **master** 分支上提交更改。

需要提交的文件修改通通放到暂存区，然后，一次性提交暂存区的所有修改。

我们再练习一遍，先对 **readme.docx** 做个修改，比如加上一行内容：

2022 年 6 月 1 日星期三

多云，今天是六一儿童节，又是开心的一天呢。

2022 年 6 月 2 日星期四

中雨，今天是农历五月初四，明天就是端午节了。

2022 年 6 月 3 日星期五

中雨，今天是农历五月初五，是中国传统节日：端午节，这一天我们要吃粽子，赛龙舟。

2022 年 6 月 7 日星期二

晴，今天是高考第一天，上午考语文，下午考数学。

然后，在工作区新增一个 `license.txt` 的文本文件。



先用 `git status` 查看一下状态：

```
李响1@Lixiang MINGW64 /d/myrepo (master)
$ git status
On branch master
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:   readme.docx

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

no changes added to commit (use "git add" and/or "git commit -a")
```

Git 非常清楚地告诉我们，`readme.txt` 被修改了，而 `license.txt` 还从来没有被添加过，所以它的状态是 `Untracked`。

现在，使用两次命令 `git add`，把 `readme.txt` 和 `license.txt` 都进行添加：

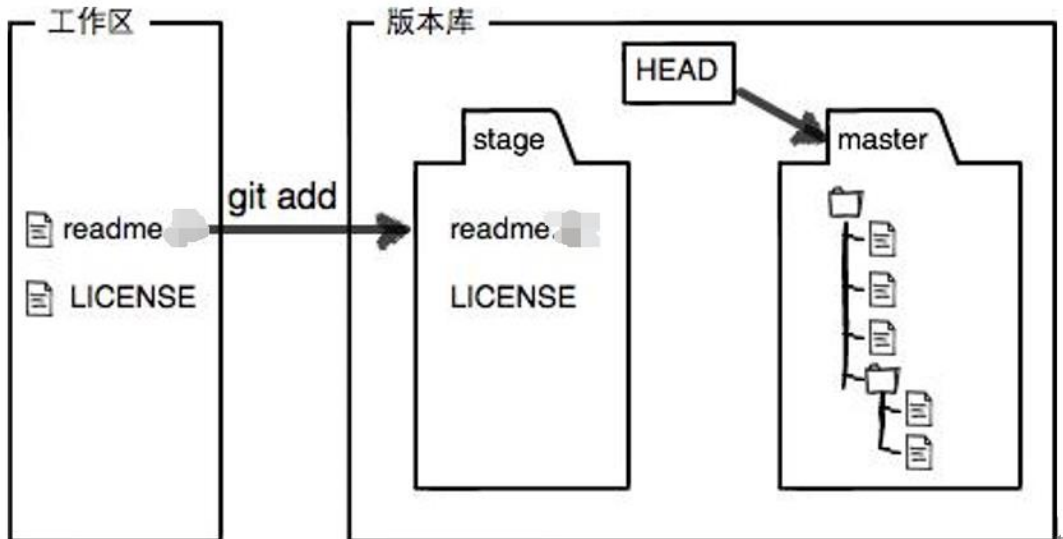
```
李响1@Lixiang MINGW64 /d/myrepo (master)
$ git add readme.docx

李响1@Lixiang MINGW64 /d/myrepo (master)
$ git add license.txt
```

再用 `git status` 再查看一下：

```
李响1@LiXiang MINGW64 /d/myrepo (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   license.txt
    modified:   readme.docx
```

现在，暂存区的状态就变成这样了：



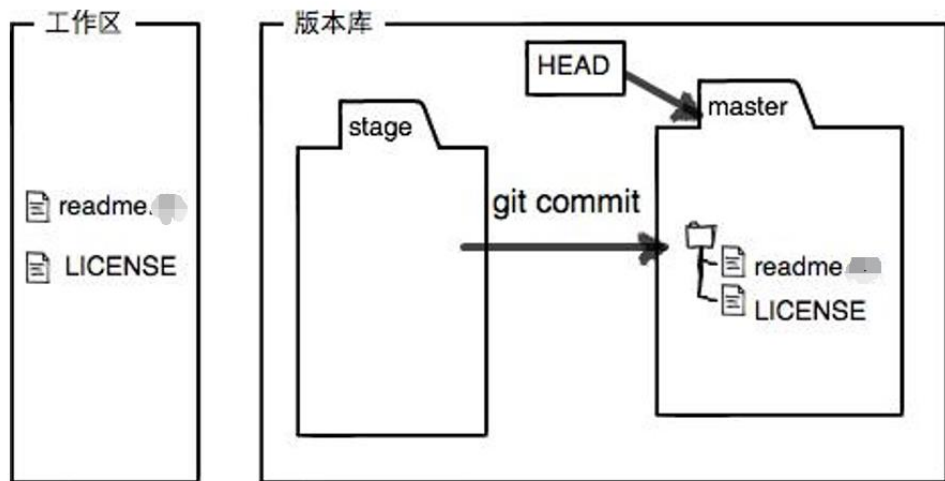
所以，`git add` 命令实际上就是把要提交的所有修改放到暂存区（Stage），然后，执行 `git commit` 就可以一次性把暂存区的所有修改提交到分支。

```
李响1@LiXiang MINGW64 /d/myrepo (master)
$ git commit -m "understand how stage works"
[master 4818645] understand how stage works
2 files changed, 1 insertion(+)
create mode 100644 license.txt
```

一旦提交后，如果你又没有对工作区做任何修改，那么工作区就是“干净”的，再次执行 `git status` 命令，结果如下：

```
李响1@LiXiang MINGW64 /d/myrepo (master)
$ git status
On branch master
nothing to commit, working tree clean
```

现在版本库变成了这样，暂存区 `stage` 就提交到了 `master`，和 `master` 就没有差别，没有任何要提交的内容了：



总结：

Git 管理的文件分为：工作区，版本库，版本库又分为暂存区 **stage** 和暂存区分支 **master**(仓库)

工作区>>>>暂存区>>>>仓库

**git add** 把文件从工作区>>>>暂存区，**git commit** 把文件从暂存区>>>>仓库，

**git diff** 查看工作区和暂存区差异，**git diff --cached** 查看暂存区和仓库差异，

**git diff HEAD** 查看工作区和仓库的差异，

**git add** 的反向命令 **git checkout**，撤销工作区修改，即把暂存区最新版本转移到工作区，

**git commit** 的反向命令 **git reset HEAD**，就是把仓库最新版本转移到暂存区。

**git status** 查看工作区状态

## 步骤 2：管理修改

### 1、Git 跟踪并管理的是修改，而非文件。

什么是修改？比如你新增了一行，这就是一个修改，删除了一行，也是一个修改，更改了某些字符，也是一个修改，删了一些又加了一些，也是一个修改，甚至创建一个新文件，也算一个修改。

为什么说 **Git** 管理的是修改，而不是文件呢？我们还是做实验。第一步，对 **readme.docx** 做一个修改，比如增加了内容：“今天天气不错，”

2022 年 6 月 1 日星期三

多云，今天是六一儿童节，又是开心的一天呢。

2022 年 6 月 2 日星期四

中雨，今天是农历五月初四，明天就是端午节了。

2022 年 6 月 3 日星期五

中雨，今天是农历五月初五，是中国传统节日：端午节，这一天我们要吃粽子，赛龙舟。

2022 年 6 月 7 日星期二

晴，今天是高考第一天，上午考语文，下午考数学。今天天气不错，

然后，执行 git add 命令：

```
李响1@Lixiang MINGW64 /d/myrepo (master)
$ git add readme.docx
```

在执行查看工作区状态 git status 命令：

```
李响1@Lixiang MINGW64 /d/myrepo (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   readme.docx
```

然后，再次修改 readme.docx：

2022 年 6 月 1 日星期三

多云，今天是六一儿童节，又是开心的一天呢。

2022 年 6 月 2 日星期四

中雨，今天是农历五月初四，明天就是端午节了。

2022 年 6 月 3 日星期五

中雨，今天是农历五月初五，是中国传统节日：端午节，这一天我们要吃粽子，赛龙舟。

2022 年 6 月 7 日星期二

晴，今天是高考第一天，上午考语文，下午考数学。今天天气不错，心情也很好。

执行 git commit 命令提交：

```
李响1@Lixiang MINGW64 /d/myrepo (master)
$ git commit -m "add words in June 7th"
[master dec163e] add words in June 7th
1 file changed, 0 insertions(+), 0 deletions(-)
```

提交后，再次使用 git status 命令查看状态：

```
李响1@Lixiang MINGW64 /d/myrepo (master)
$ git status
On branch master
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:   readme.docx

no changes added to commit (use "git add" and/or "git commit -a")
```

发现问题：第二次的修改没有被提交！！！！

回顾一下我们的操作过程：

第一次修改 -> git add -> 第二次修改 -> git commit

Git 管理的是修改，当你用 `git add` 命令后，在工作区的第一次修改被放入暂存区，准备提交，但是，在工作区的第二次修改并没有放入暂存区，所以，`git commit` 只负责把暂存区的修改提交了，也就是第一次的修改被提交了，第二次的修改不会被提交。

我们使用 `git diff HEAD -- readme.docx` 命令可以查看工作区和版本库里面最新版本的差别：

```
李响1@Lixiang MINGW64 /d/myrepo (master)
$ git diff HEAD -- readme.docx
diff --git a/readme.docx b/readme.docx
index 0707a6f..c137544 100644
--- a/readme.docx
+++ b/readme.docx
@@ -5,4 +5,4 @@
 2022年6月3日星期五
 中雨，今天是农历五月初五，是中国传统节日：端午节，这一天我们要吃粽子，赛龙舟。
 2022年6月7日星期二
-晴，今天是高考第一天，上午考语文，下午考数学。今天天气不错，
+晴，今天是高考第一天，上午考语文，下午考数学。今天天气不错，心情也很好。
```

可见，第二次修改确实没有被提交。

那怎么提交第二次修改呢？你可以继续 `git add` 再 `git commit`，也可以别着急提交第一次修改，先 `git add` 第二次修改，再 `git commit`，就相当于把两次修改合并后一块提交了：

第一次修改 -> `git add` -> 第二次修改 -> `git add` -> `git commit`

我们把第二次修改提交了：

```
李响1@Lixiang MINGW64 /d/myrepo (master)
$ git add readme.docx

李响1@Lixiang MINGW64 /d/myrepo (master)
$ git commit -m "add words in June 7th second"
[master fb31c6e] add words in June 7th second
1 file changed, 0 insertions(+), 0 deletions(-)

李响1@Lixiang MINGW64 /d/myrepo (master)
$ git status
On branch master
nothing to commit, working tree clean

李响1@Lixiang MINGW64 /d/myrepo (master)
$ git diff HEAD -- readme.docx

李响1@Lixiang MINGW64 /d/myrepo (master)
$ |
```

总结：

每次修改，如果不用 `git add` 到暂存区，那就不会加入到 `commit` 中。



## 步骤 3: 撤销修改

1、当你改乱了工作区某个文件的内容，比如手一抖，误删了几个字。【误删】

2022 年 6 月 1 日星期三

多云，今天是六一儿童节，又是开心的一天呢。

2022 年 6 月 2 日星期四

中雨，今天是农历五月初四，明天就是端午节了。

2022 年 6 月 3 日星期五

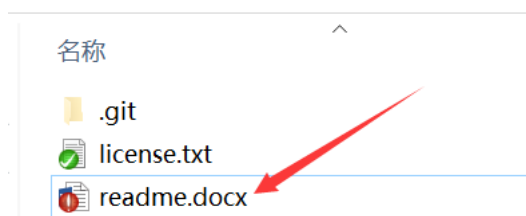
中雨，今天是农历五月初五，是中国传统节日：端午节，这一天我们要吃粽子，赛龙舟。

2022 年 6 月 7 日星期二

晴，今天是高考第一天，上午考语文，下午考数学。今很好。

显示文件已经被修改：

电脑 > 新加卷 (D:) > myrepo

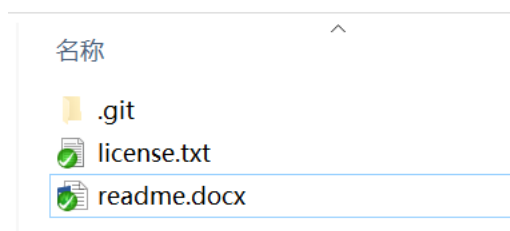


那么，我们用命令 `git checkout -- filename` 来直接丢弃工作区的修改：

```
李响1@Lixiang MINGW64 /d/myrepo (master)
$ git checkout -- readme.docx
```

执行完命令后，我们发现文件图标恢复了一个绿色的勾：

电脑 > 新加卷 (D:) > myrepo



误删的文字又回来了：

2022 年 6 月 1 日星期三

多云，今天是六一儿童节，又是开心的一天呢。

2022 年 6 月 2 日星期四

中雨，今天是农历五月初四，明天就是端午节了。

2022 年 6 月 3 日星期五

中雨，今天是农历五月初五，是中国传统节日：端午节，这一天我们要吃粽子，赛龙舟。

2022 年 6 月 7 日星期二

晴，今天是高考第一天，上午考语文，下午考数学。今天天气不错，心情也很好。



2、当你不但改乱了工作区某个文件的内容：【误删并 git add 到了暂存区】

2022 年 6 月 1 日星期三

多云，今天是六一儿童节，又是开心的一天呢。

2022 年 6 月 2 日星期四

中雨，今天是农历五月初四，明天就是端午节了。

2022 年 6 月 3 日星期五

中雨，今天是农历五月初五，是中国传统节日：端午节，这一天我们要吃粽子，赛龙舟。

2022 年 6 月 7 日星期二

晴，今天是高考第一天，上午考语文，下午考数学。今很好。

还使用 git add 添加到了暂存区：

```
李响1@Lixiang MINGW64 /d/myrepo (master)
$ git add readme.docx

李响1@Lixiang MINGW64 /d/myrepo (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   readme.docx
```

这时想丢弃修改，分两步，第一步用命令 git reset HEAD filename 可以把暂存区的修改撤销掉（unstage），重新放回工作区：

```
李响1@Lixiang MINGW64 /d/myrepo (master)
$ git reset HEAD readme.docx
Unstaged changes after reset:
M       readme.docx
```

第二步按 1 操作，用命令 git checkout -- filename 来直接丢弃工作区的修改：

```
李响1@Lixiang MINGW64 /d/myrepo (master)
$ git checkout -- readme.docx
```

电脑 > 新加卷 (D:) > myrepo

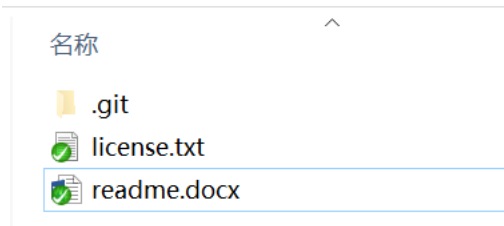
名称

.git

license.txt

readme.docx

执行完命令后，我们发现文件图标恢复了一个绿色的勾：



误删并已经 `git add` 放到暂存区的文字又恢复了：

```
2022 年 6 月 1 日星期三
多云，今天是六一儿童节，又是开心的一天呢。
2022 年 6 月 2 日星期四
中雨，今天是农历五月初四，明天就是端午节了。
2022 年 6 月 3 日星期五
中雨，今天是农历五月初五，是中国传统节日：端午节，这一天我们要吃粽子，赛龙舟。
2022 年 6 月 7 日星期二
晴，今天是高考第一天，上午考语文，下午考数学。今天天气不错，心情也很好。
```

3、已经提交了不合适的修改到版本库时，想要撤销本次提交，参考版本回退一节，不过前提是没有推送到远程库。

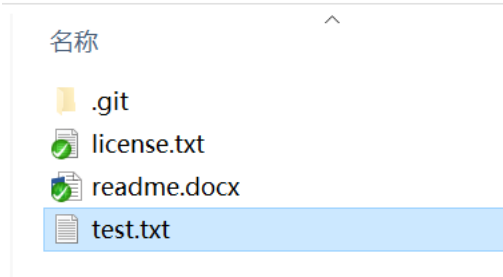
#### 4、新版命令

- 从暂存区恢复工作区：`git restore --worktree readme.docx`
- 从 master 恢复暂存区：`git restore --staged readme.docx`
- 从 master 同时恢复工作区和暂存区：  
`git restore --source=HEAD --staged --worktree readme.docx`

## 步骤 4：删除文件

在 Git 中，删除也是一个修改操作，我们先在 myrepo 文件夹中添加一个新文件 test.txt：

电脑 > 新加卷 (D:) > myrepo >

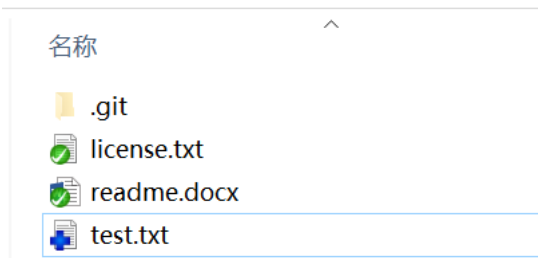


使用 git add 命令添加到 Git 库中：

```
李响1@Lixiang MINGW64 /d/myrepo (master)
$ git add test.txt
```

再次观察 myrepo 文件夹中的文件：（test.txt 带了一个蓝色的加号）

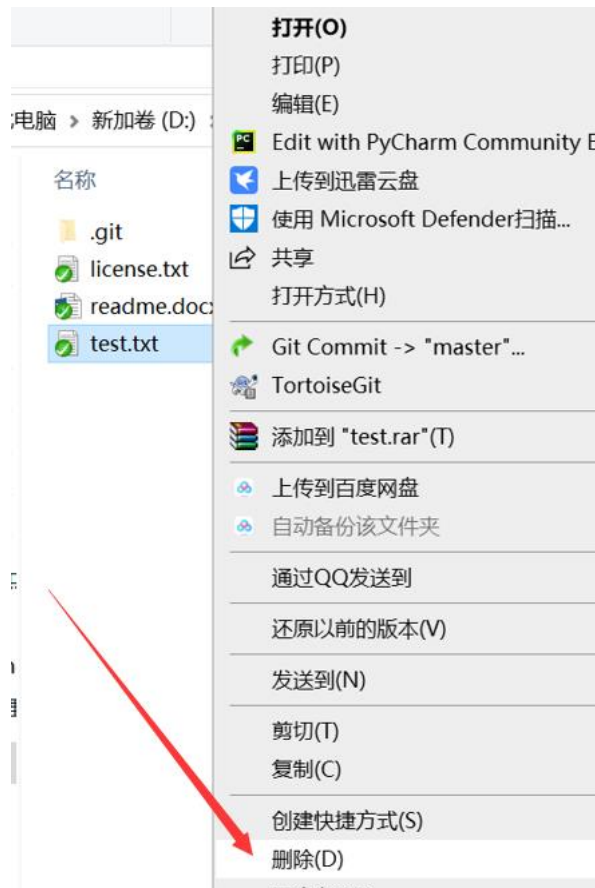
电脑 > 新加卷 (D:) > myrepo



使用 git commit 命令提交文件：

```
李响1@Lixiang MINGW64 /d/myrepo (master)
$ git commit -m "add test.txt"
[master e1cbd59] add test.txt
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 test.txt
```

接下来，我们不想要这个文件了，我们把 test.txt 从 myrepo 文件夹中手动删除：



这时，Git 知道你删除了文件，因此，工作区和版本库就不一致了，git status 命令会告诉你哪些文件被删除了：

```
李响1@Lixiang MINGW64 /d/myrepo (master)
$ git status
on branch master
Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
    deleted:    test.txt

no changes added to commit (use "git add" and/or "git commit -a")
```

现在你有两个选择，一是确实要从版本库中删除该文件，那就用命令 git rm 删掉，并且 git commit：

```
李响1@Lixiang MINGW64 /d/myrepo (master)
$ git rm test.txt
rm 'test.txt'

李响1@Lixiang MINGW64 /d/myrepo (master)
$ git commit -m "remove test.txt"
[master dbf9ce6] remove test.txt
1 file changed, 0 insertions(+), 0 deletions(-)
delete mode 100644 test.txt
```

另一种情况是删错了，因为版本库里还有呢，所以可以很轻松地把误删的文件恢复到最新版本：（使用新版本的 `git restore` 命令）

```
李响1@LiXiang MINGW64 /d/myrepo (master)
$ git restore test.txt


李响1@LiXiang MINGW64 /d/myrepo (master)
$ git status
On branch master
nothing to commit, working tree clean
```


查看 myrepo 目录，发现删除的文件已经恢复了：


电脑 > 新加卷 (D:) > myrepo

名称 ^

 .git

 license.txt

 readme.docx

 test.txt