前言

前两篇Git笔记主要内容是Git的安装以及底层命令，但是在实际开发当中是几乎不涉及底层命令的，因为底层命令记忆麻烦而且容易出错。既然实际开发过程不用底层命令，那为啥还要学习呢？emmmm好问题，底层命令可以说是Git的灵魂高层命令是Git的身躯，单纯学Git的高层命令那是没有灵魂de~不是有一首歌的歌词是“得到你的人却得不到你的心，就是得到全世界也不开心~~~~”,Git也是如此的耶！既然我们已经有了Git的灵魂，那么我们是时候开始获得Git的身躯了。

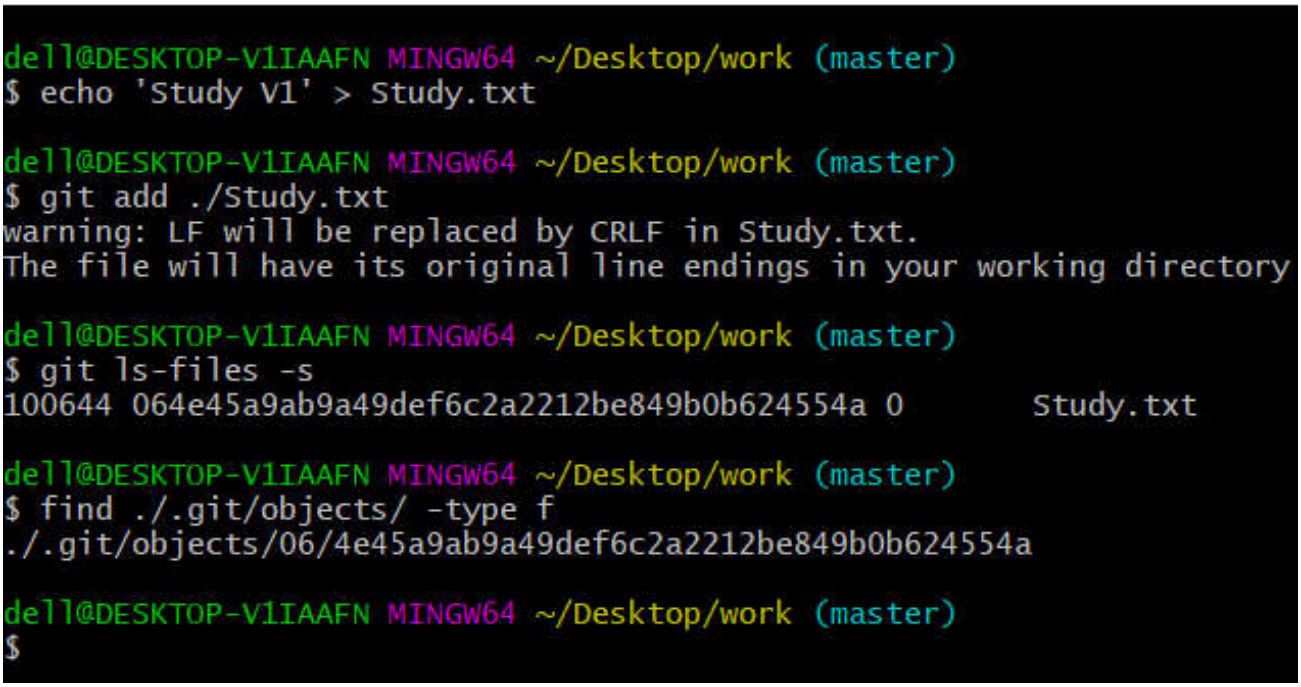
一、高层命令

初始化仓库

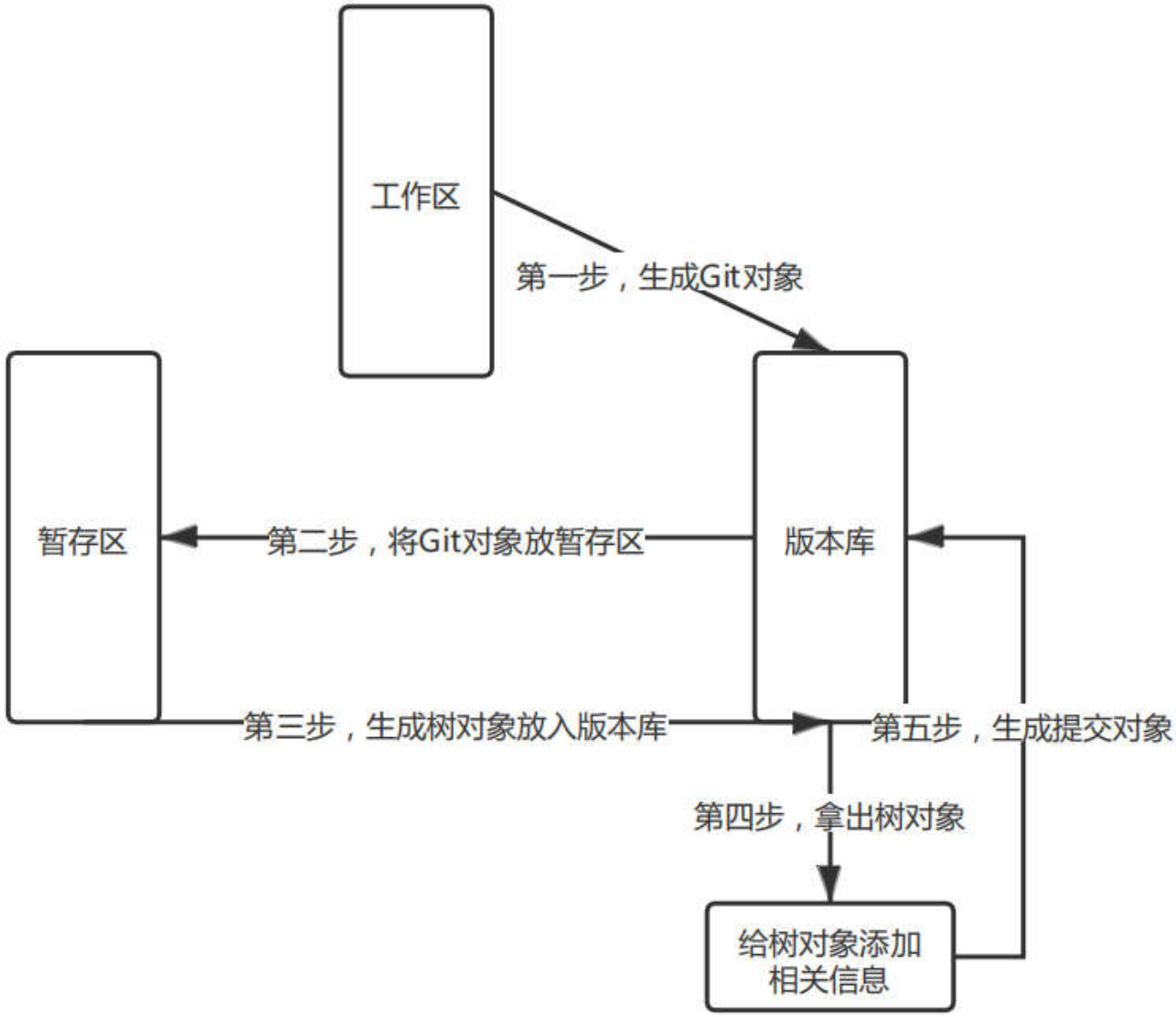
- **git init**
  这个命令在前面已经使用过，就是用来初始化Git仓库的。

将修改添加进暂存区

- **git add 文件路径**
  光说不练假把式，所以我们应该动手试一试。流程：创建一个文件，然后利用git add 命令将文件加入暂存区、 git ls-files -s 命令查看暂存区内容 和 find ./.git/objects/ -type f 命令查看版本库内容。



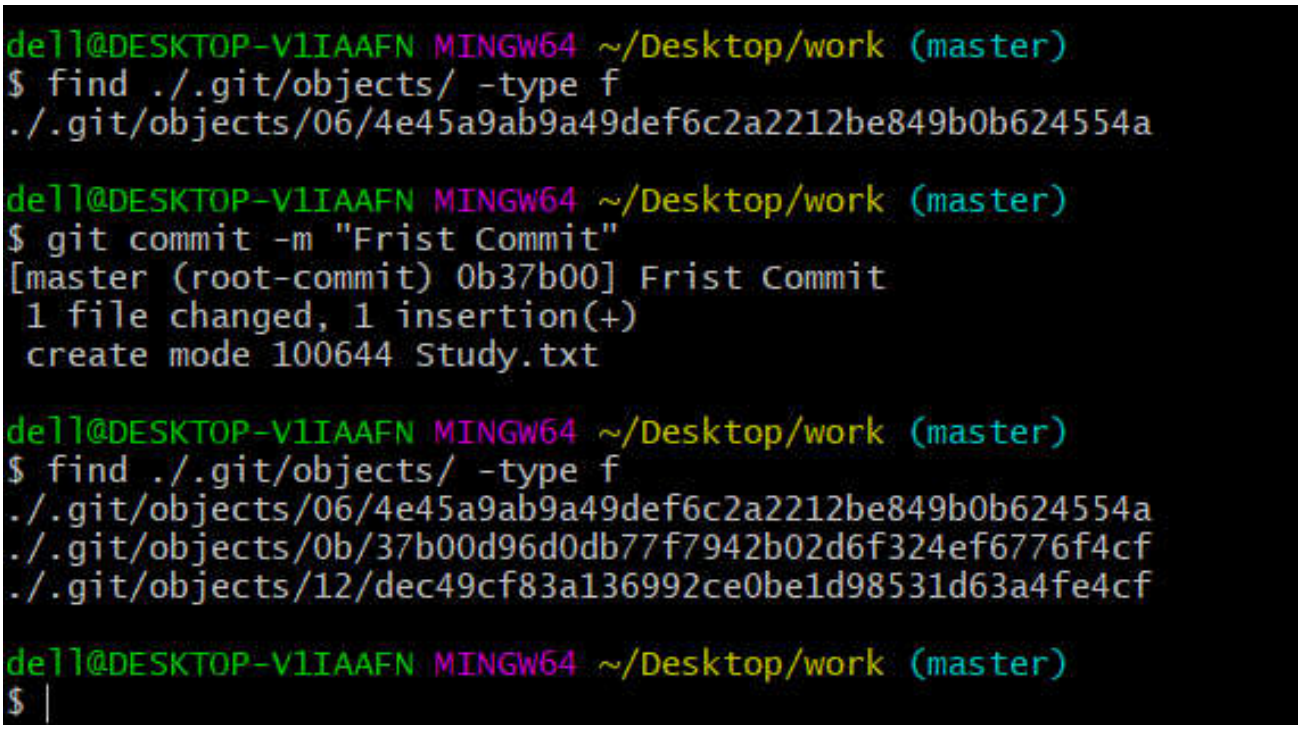
我在Git学习笔记二中写道，Git的文件管理是从工作区到暂存区最好后在进入版本库，但是从上图就可以发现我以前说的是不对的,我们可以看见现在暂存区和版本库里面都有。所以，那么真正的流程应该是怎么样的呢？下面我用一个图来进行描述。



所以通过上面的我们可以分析出git add 命令应该就是 git hash-object -w 和 git updata-index的结合体，看到这里我相信大家的思路应该是异常的清晰吧哈哈。

将暂存区的内容提交到版本库

- **git commit -m "infromation"**
- **git commit "** 可以在vim编辑器中写大量的信息
- **git commit -a -m "infromation"** 跳过暂存区直接进入版本库(其实是让Git自己执行了git add的命令)
  git commit 命令应该就是git write-tree 和 git commit-tree的结合体，废话不多说直接上图。(我肯定演示最简单的哈哈哈，剩下的给你们去实践)



没用使用git commit的命令时候版本库里面只有一个对象，当我们使用完命令之后就变成三个对象，这三个对象分别是git对象、树对象以及提交对象。

检测当前文件状态

- **git status**

  首先，我们需要知道文件有两大状态分别是**已跟踪和未跟踪**，已跟踪又分为**已暂存**、**已提交和已修改**。

```
dell@DESKTOP-V1IAAFN MINGW64 ~/Desktop/work
$ git init
Initialized empty Git repository in C:/Users/dell/Desktop/work/.git/

dell@DESKTOP-V1IAAFN MINGW64 ~/Desktop/work (master)
$ echo 'new v1' > new.txt

dell@DESKTOP-V1IAAFN MINGW64 ~/Desktop/work (master)
$ git status
On branch master

No commits yet

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

nothing added to commit but untracked files present (use "git add" to track)

dell@DESKTOP-V1IAAFN MINGW64 ~/Desktop/work (master)
$ git add ./new.txt
warning: LF will be replaced by CRLF in new.txt.
The file will have its original line endings in your working directory

dell@DESKTOP-V1IAAFN MINGW64 ~/Desktop/work (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   new.txt


dell@DESKTOP-V1IAAFN MINGW64 ~/Desktop/work (master)
$ gim commit -m "First Commit"
bash: gim: command not found

dell@DESKTOP-V1IAAFN MINGW64 ~/Desktop/work (master)
$ git commit -m "First Commit"
[master (root-commit) 7f1adac] First Commit
 1 file changed, 1 insertion(+)
 create mode 100644 new.txt

dell@DESKTOP-V1IAAFN MINGW64 ~/Desktop/work (master)
$ git status
On branch master
nothing to commit, working tree clean
```



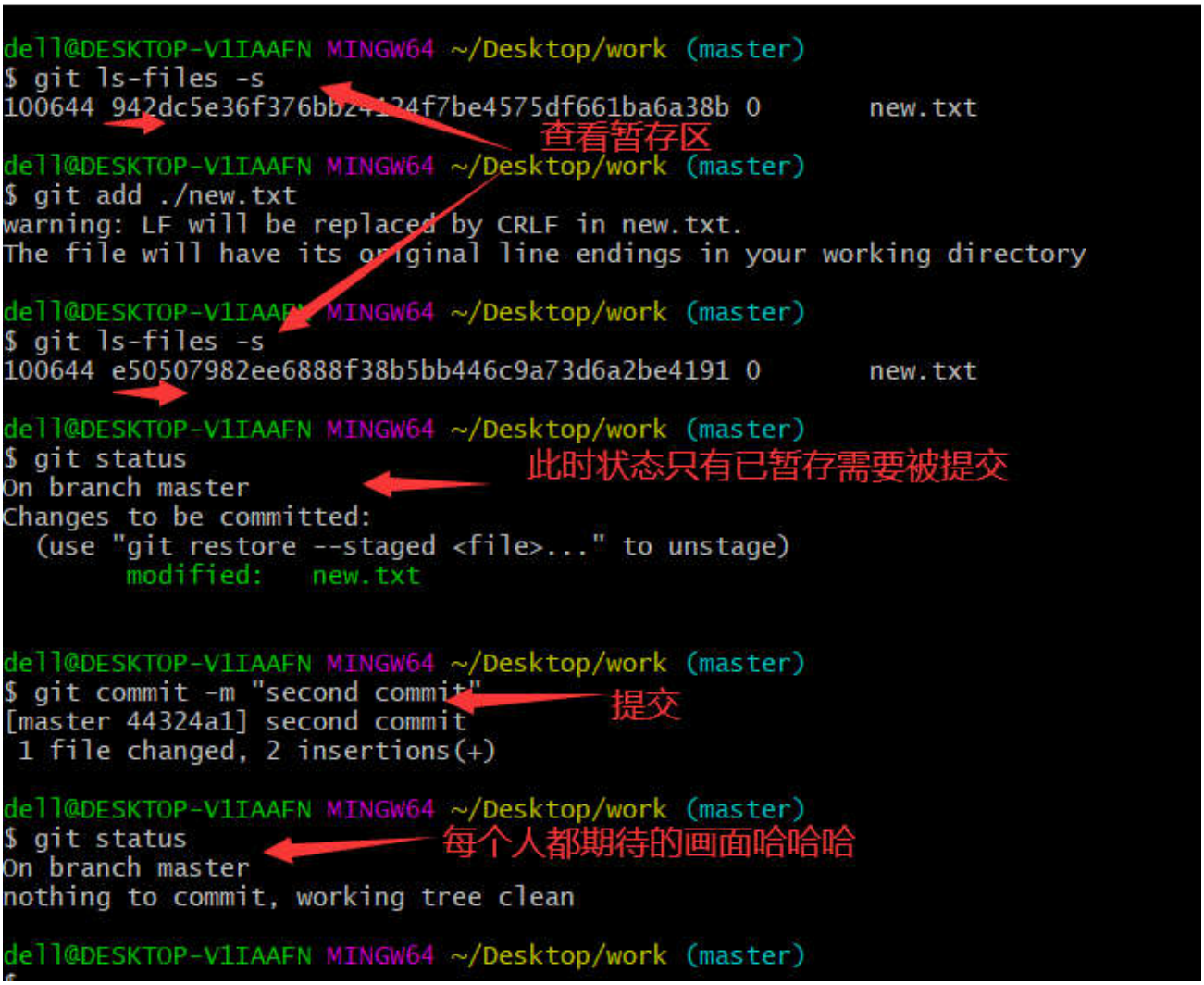
```
dell@DESKTOP-V1IAAFN MINGW64 ~/Desktop/work (master)
$ vim new.txt

dell@DESKTOP-V1IAAFN MINGW64 ~/Desktop/work (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:   new.txt

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

文件的状态不同我们所采取的行动也不同，文件未跟踪那么我们需要跟踪一下，文件若处于已修改那么我们应该暂存一下，文件处于已暂存那么我们需要提交一下，提交完就可以安心睡大觉了(狗头保命哈哈)。上述情况是比较理想的，现在我们想一种情况：一个文件已经暂存了，但是我现在又修改了这个文件那么git status会提示什么信息呢？废话不多说直接上图！(希望初学Git可以自己尝试的哦！)

```
dell@DESKTOP-V1IAAFN MINGW64 ~/Desktop/work (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   new.txt


dell@DESKTOP-V1IAAFN MINGW64 ~/Desktop/work (master)
$ vim new.txt

dell@DESKTOP-V1IAAFN MINGW64 ~/Desktop/work (master)
$ git stuats
git: 'stuats' is not a git command. See 'git --help'.

The most similar command is
        status

dell@DESKTOP-V1IAAFN MINGW64 ~/Desktop/work (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   new.txt

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:   new.txt


dell@DESKTOP-V1IAAFN MINGW64 ~/Desktop/work (master)
$
```

在图中的提示可以看见提示文件已暂存需要提交和已修改需要暂存，这样就由问题如果我们此时执行git commit -m 那我们刚才修改会被提交上去吗？答案肯定是不会的，所以我们需要先在次执行git add 命令将修改的内容进行暂存,因为进入暂存区的内容和原来的暂存区具有相同的文件名所以会出现覆盖(Ps:这个覆盖原来这个更加人性化，但是本质不会覆盖因为内容不同生成的hash值是不同的，如果忘记了记得翻一下我以前

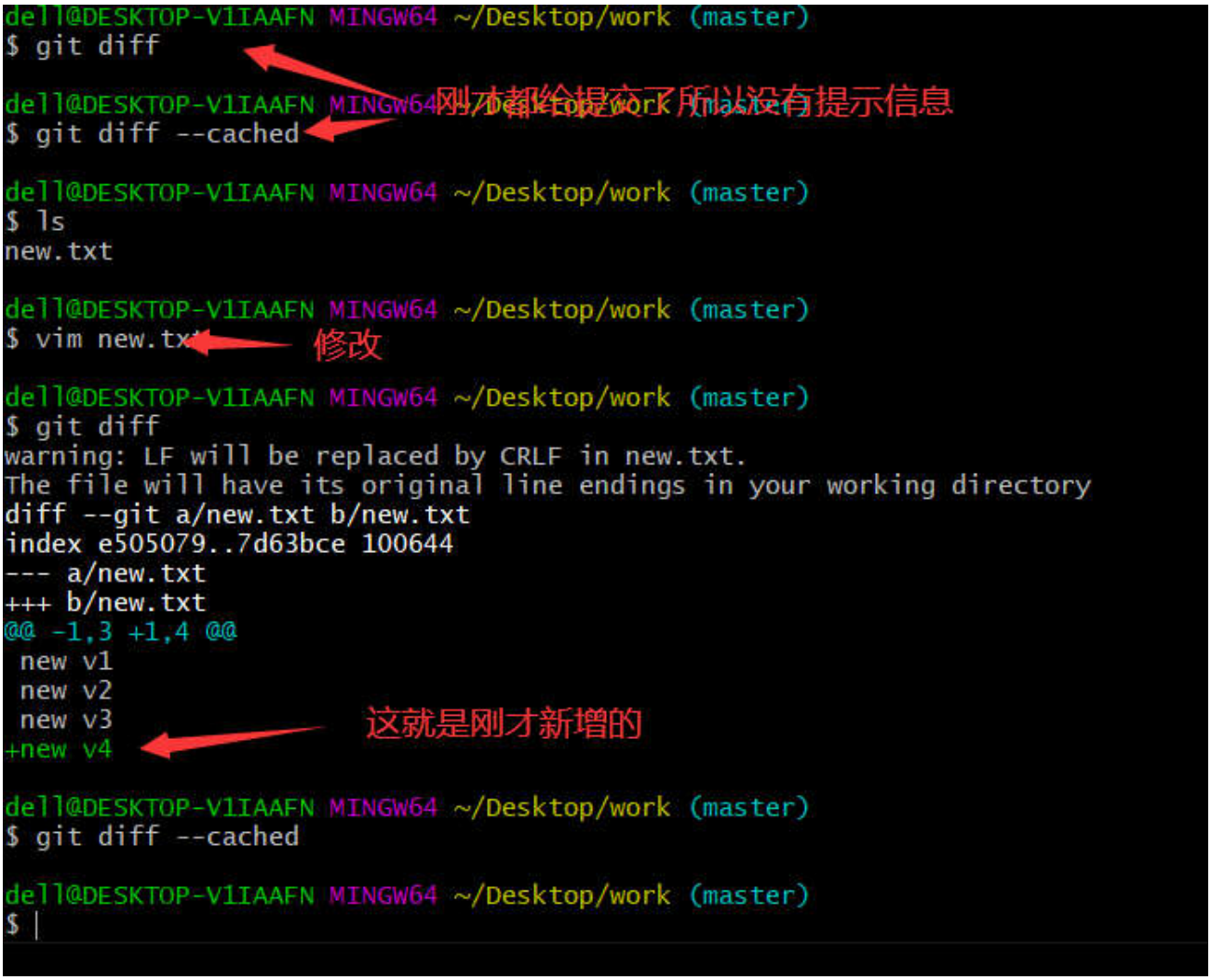
的笔记哦！)废话不多说上图咯！



检测已暂存和未暂存的更新

- **git diff**
- **git diff --cached**

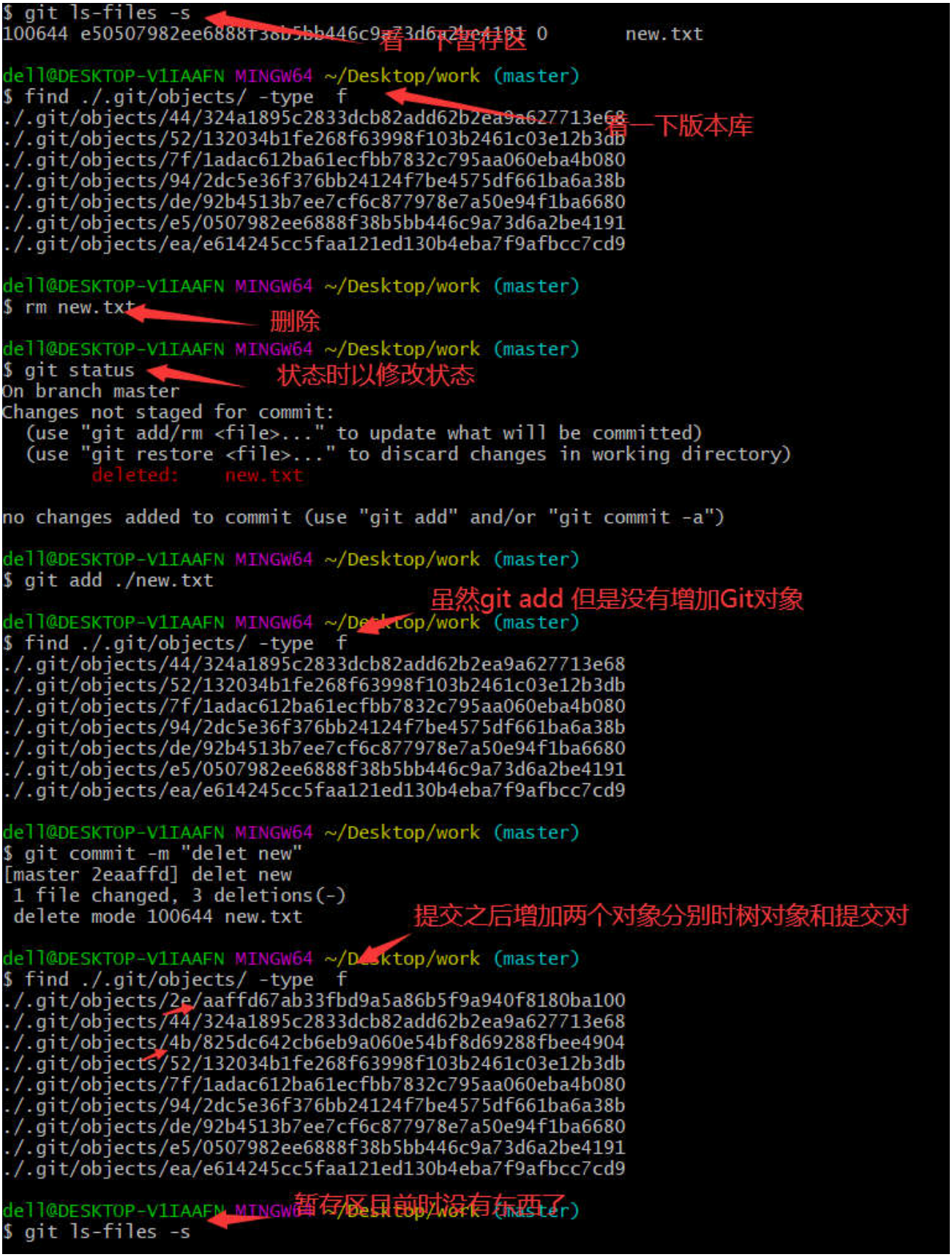
  对于git status来言，git diff可以让我们看到更多细节可以帮助我们查看具体修改那些内容。git diff 命令：当前的操作是不是修改了但是没有保存；gif diff --cached 命令那些东西暂存了但是没有提交。(Ps:记得自己尝试哦！)



删除文件

- **rm 文件路径**
- **git rm 文件路径**(省去了我们再出输入git add命令)

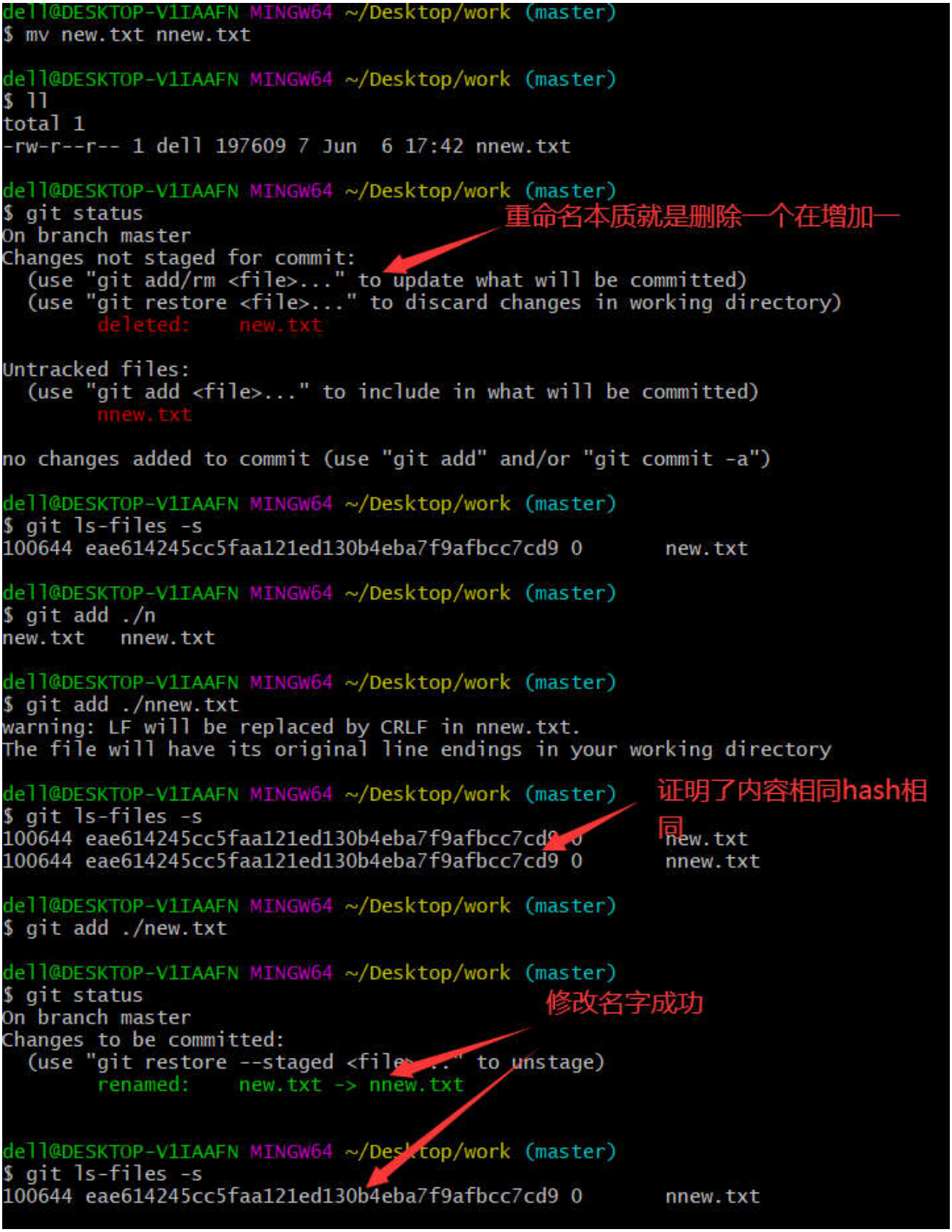
  从Git移除文件，那肯定要在已跟踪的文件清单进行删除，通过上面的实验我们知道当执行了git add之后文件就会被跟踪所以我们需要在暂存区将其删除。Git肯定是不会给我们删除的，删除只是表面时看起来删除了本质上还是增加操作，直接上图(图中演示时用的rm)。



浮一下思路，删除操作也就是把工作目录中的new.txt删除,暂存区中new.txt的快照移除，然后将此时暂存区的快照生成一个树对象,然后再通过git commit 生成提交对象。所以我们的版本库里面会增加两个对象一个树对象和一个提交对象。

文件重命名

- **mv fileName newFileName**
- **git mv fileName newFileName**(省去了我们再出输入git add命令)
  git mv 将其分解其实是三个命令 mv name newName;git rm name;git add newName;直接上图(图中演示的时mv)



查看日志

- **git log --oneline**

  这个没啥好说的就是看相关日志，这个大家自己尝试一下就ok

啦SO~~easy！

二、 结尾

至此呢Git的经常使用的高层命令就讲述完了下一次就是讲述Git最最最最牛皮的分支功能哈哈哈，如果感觉有用可以点个赞的哦！我会持续更新，如果有错误还请指出来,感谢观众老爷的赏脸。

若想获得上述内容的PDF版本移步到GitHub下载。

地址：Git 学习笔记专区。