

左边为git的某个版本号的代码，中间为左边的代码的base版本，右边的是自己提交过的版本（这个待定，也可能是stash的版本？）

|  |
| --- |
| public class CommonAnnounceAdapter extends RecyclerView.Adapter<CommonAnnounceAdapter.CommonAnnounceHolder> |
| implements View.OnClickListener { |
|  |
| public List<CommonNoticeOrder> entities = new ArrayList<>(); |

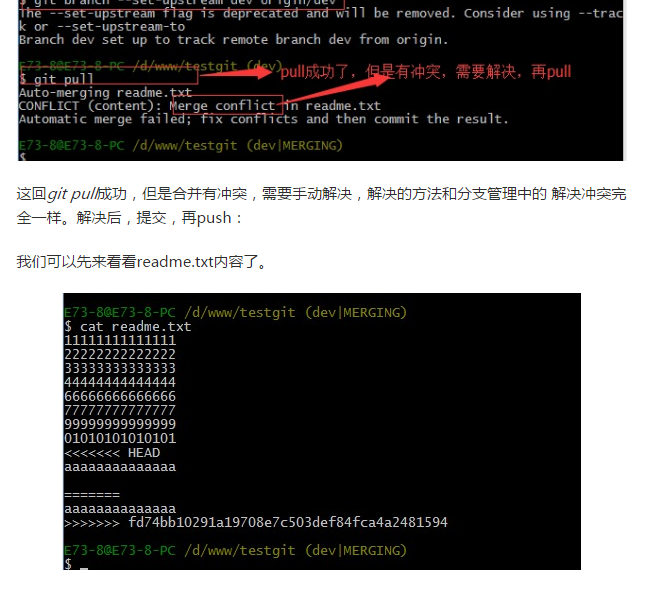
数据集加final

**工作区、版本库、暂存区原理图**

在这个图中，我们可以看到部分 Git 命令是如何影响工作区和暂存区（stage, index）的。

* 图中左侧为工作区，右侧为版本库。在版本库中标记为 “index” 的区域是暂存区（stage, index），标记为 “master” 的是 master 分支所代表的目录树。
* 图中我们可以看出此时 “HEAD” 实际是指向 master 分支的一个“游标”。所以图示的命令中出现 HEAD 的地方可以用 master 来替换。
* 图中的 objects 标识的区域为 Git 的对象库，实际位于 “.git/objects” 目录下，我们会在后面的章节重点介绍。
* 当对工作区修改（或新增）的文件执行 “git add” 命令时，暂存区的目录树被更新，同时工作区修改（或新增）的文件内容被写入到对象库中的一个新的对象中，而该对象的ID 被记录在暂存区的文件索引中。
* 当执行提交操作（git commit）时，暂存区的目录树写到版本库（对象库）中，master 分支会做相应的更新。即 master 指向的目录树就是提交时暂存区的目录树。
* 当执行 “git reset HEAD” 命令时，暂存区的目录树会被重写，被 master 分支指向的目录树所替换，但是工作区不受影响。
* 当执行 “git rm –cached <file>” 命令时，会直接从暂存区删除文件，工作区则不做出改变。
* 当执行 “git checkout .” 或者 “git checkout — <file>” 命令时，会用暂存区全部或指定的文件替换工作区的文件。这个操作很危险，会清除工作区中未添加到暂存区的改动。
* 当执行 “git checkout HEAD .” 或者 “git checkout HEAD <file>” 命令时，会用 HEAD 指向的 master 分支中的全部或者部分文件替换暂存区和以及工作区中的文件。这个命令也是极具危险性的，因为不但会清除工作区中未提交的改动，也会清除暂存区中未提交的改 动。





因此：多人协作工作模式一般是这样的：

1. 首先，可以试图用git push origin branch-name推送自己的修改.
2. 如果推送失败，则因为远程分支比你的本地更新早，需要先用git pull试图合并。
3. 如果合并有冲突，则需要解决冲突，并在本地提交。再用git push origin branch-name推送。

1. 防止被引用的lib中使用的库的版本和主工程不同

compile(project(‘:my-library‘)){

transitive=false;

}

2. grale 的android 标签下有个defaultPublishConfig 的属性，是用来控制发布的lib是release还是debug版本，如果有flavor的话，还可以根据flavor来生成不通的变种版本