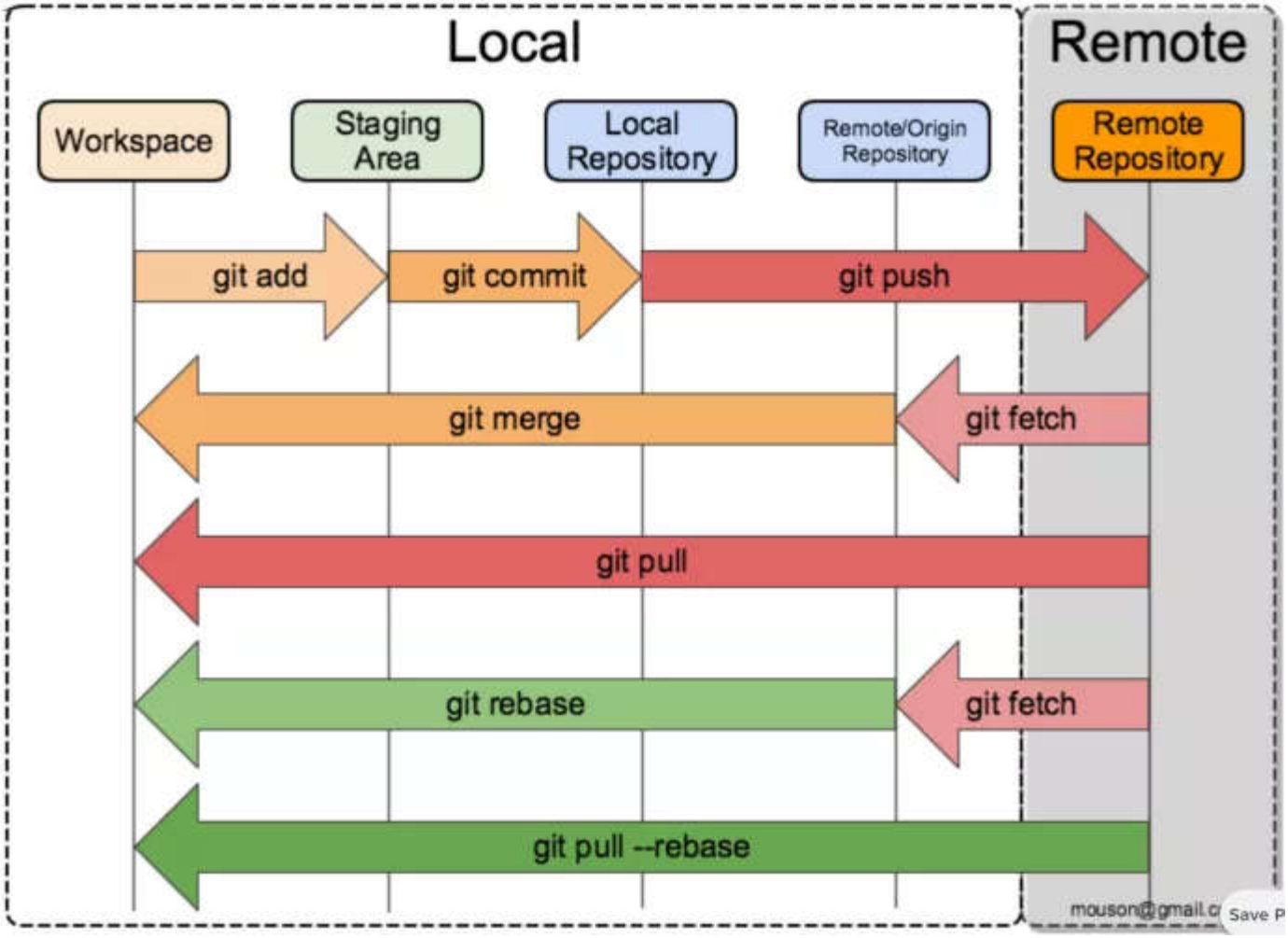
Git 的 4 个阶段的撤销更改

2017-12-03 Linux爱好者

(点击上方蓝字，快速关注我们)

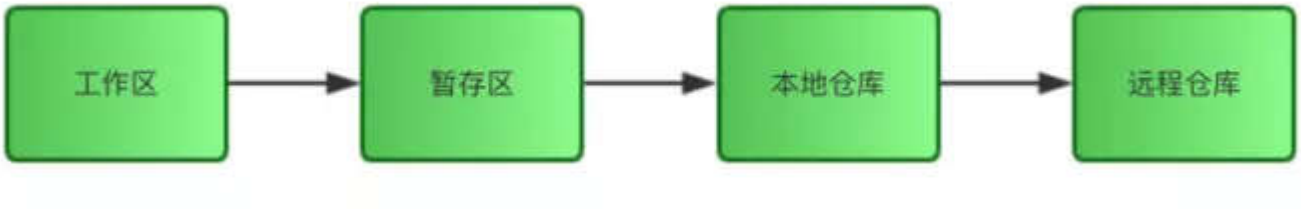
> 来源：张京
> www.fengerzh.com/git-reset/



虽然git诞生距今已有12年之久，网上各种关于git的介绍文章数不胜数，但是依然有很多人（包括我自己在内）对于它的功能不能完全掌握。以下的介绍只是基于我个人对于git的理解，并且可能生编硬造了一些不完全符合git说法的词语。目的只是为了让git通俗化，使初学者也能大概了解如何快速上手git。同时，下面所有讨论，我们都假设只使用一个分支，也就是主分支master的情况，虽然这种作法并不符合git规范，但是现实情况中绝大部分用户是直接在master分支上进行工作的，所以在这里我们不去引入更加复杂的各种分支的情况，也不涉及标签tag的操作，只讲在最简单的主分支上如何回退。

基本概念

3个步骤



正常情况下，我们的工作流就是3个步骤，对应上图中的3个箭头线：

```
git add .
git commit -m "comment"
git push
```

1. git add .把所有文件放入暂存区；
2. git commit把所有文件从暂存区提交进本地仓库；
3. git push把所有文件从本地仓库推送进远程仓库。

4个区

git之所以令人费解，主要是它相比于svn等等传统的版本管理工具，多引入了一个暂存区(Stage)的概念，就因为多了这一个概念，而使很多人疑惑。其实，在初学者来说，每个区具体怎么工作的，我们完全不需要关心，而只要知道有这么4个区就够了：

- 工作区(Working Area)
- 暂存区(Stage)
- 本地仓库(Local Repository)
- 远程仓库(Remote Repository)

5种状态

以上4个区，进入每一个区成功之后会产生一个状态，再加上最初始的一个状态，一共是5种状态。以下我们把这5种状态分别命名为：

- 未修改(Origin)
- 已修改(Modified)
- 已暂存(Staged)
- 已提交(Committed)
- 已推送(Pushed)

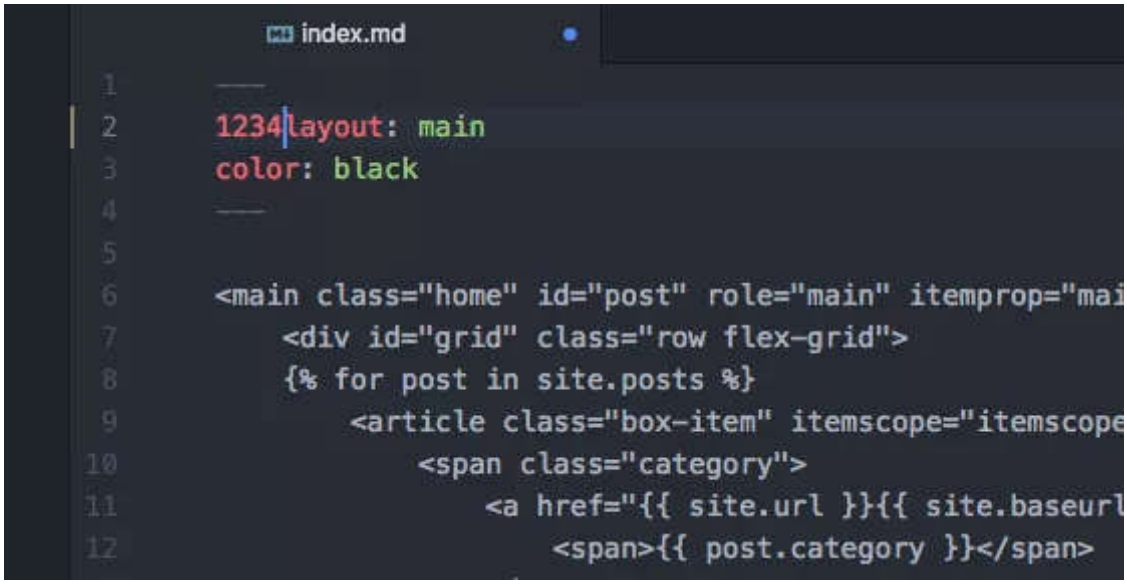
检查修改

了解了基本概念之后，我们来谈一谈犯错误之后如何撤销的问题。首先，我们要了解如何检查这3个步骤当中每一个步骤修改了什么，然后才好判断有没有修改成功。检查修改的二级命令都相同，都是diff，只是参数有所不同。

已修改，未暂存

> git diff

首先，我们来看一下，如果我们只是简单地在浏览器里保存了一下文件，但是还没有做git add .之前，我们如何检查有哪些修改。我们先随便拿一个文件来做一下实验：



我们在文件开头的第2行胡乱加了4个数字1234，存盘，这时文件进入了已修改状态，但是还没有进入暂存区，我们运行git diff，结果如下：

```
diff --git a/index.md b/index.md
index 73ff1ba..1066758 100644
--- a/index.md
+++ b/index.md
@@ -1,5 +1,5 @@
 ---
-layout: main
+1234layout: main
 color: black
 ---
```

git diff的结果告诉我们哪些文件已经做了哪些修改。

已暂存，未提交

> git diff --cached

现在我们把修改放入暂存区看一下。先执行git add .，然后执行git diff，你会发现没有任何结果：

```
zhangjing@preamble ~/P/fengerzh.github.io> git add .
zhangjing@preamble ~/P/fengerzh.github.io> git diff
zhangjing@preamble ~/P/fengerzh.github.io>
```

这说明git diff这个命令只检查我们的工作区和暂存区之间的差异，如果我们想看到暂存区和本地仓库之间的差异，就需要加一个参数git diff --cached：

```
diff --git a/index.md b/index.md
index 73ff1ba..1066758 100644
--- a/index.md
+++ b/index.md
@@ -1,5 +1,5 @@
 ---
-layout: main
+1234layout: main
 color: black
 ---
```

这时候我们看到的差异是暂存区和本地仓库之间的差异。

已提交，未推送

> git diff master origin/master

现在，我们把修改从暂存区提交到本地仓库，再看一下差异。先执行git commit，然后再执行git diff --cached，没有差异，执行git diff master origin/master，可以看到差异：

```
zhangjing@preamble ~/P/fengerzh.github.io> git commit -m "test"
[master da6b688] test
 1 file changed, 1 insertion(+), 1 deletion(-)
zhangjing@preamble ~/P/fengerzh.github.io> git diff --cached
zhangjing@preamble ~/P/fengerzh.github.io> git diff master origin/master
diff --git a/index.md b/index.md
index 1066758..73ff1ba 100644
--- a/index.md
+++ b/index.md
@@ -1,5 +1,5 @@
 ---
-1234layout: main
+layout: main
 color: black
 ---
```

在这里，master就是你的本地仓库，而origin/master就是你的远程仓库，master是主分支的意思，因为我们都在主分支上工作，所以这里两边都是master，而origin就代表远程。

撤销修改

了解清楚如何检查各种修改之后，我们开始尝试各种撤销操作。

已修改，未暂存

如果我们只是在编辑器里修改了文件，但还没有执行git add .，这时候我们的文件还在工作区，并没有进入暂存区，我们可以用：

> git checkout .

或者

> git reset --hard

来进行撤销操作。

可以看到，在执行完git checkout .之后，修改已被撤销，git diff没有任何内容了。

> 一对反义词 git add .的反义词是git checkout .。做完修改之后，如果你想向前走一步，让修改进入暂存区，就执行git add .，如果你想向后退一步，撤销刚才的修改，就执行git checkout .。

已暂存，未提交

你已经执行了git add .，但还没有执行git commit -m "comment"。这时候你意识到了错误，想要撤销，你可以执行：

> git reset
> git checkout .

或者

> git reset --hard

git reset只是把修改退回到了git add .之前的状态，也就是说文件本身还处于已修改未暂存状态，你如果想退回未修改状态，还需要执行git checkout .。

或许你已经注意到了，以上两个步骤都可以用同一个命令git reset --hard来完成。是的，就是这个强大的命令，可以一步到位地把你的修改完全恢复到未修改的状态。

已提交，未推送

你的手太快，你既执行了git add .，又执行了git commit，这时候你的代码已经进入了你的本地仓库，然而你后悔了，怎么办？不要着急，还有办法。

> git reset --hard origin/master

还是这个git reset --hard命令，只不过这次多了一个参数origin/master，正如我们上面讲过的，origin/master代表远程仓库，既然你已经污染了你的本地仓库，那么就从远程仓库把代码取回来吧。

已推送

很不幸，你的手实在是太快了，你既git add了，又git commit了，并且还git push了，这时你的代码已经进入远程仓库。如果你想恢复的话，还好，由于你的本地仓库和远程仓库是等价的，你只需要先恢复本地仓库，再强制push到远程仓库就好了：

> git reset --hard HEAD^
> git push -f

```
zhangjing@preamble ~/P/fengerzh.github.io> git reset --hard HEAD^
HEAD is now at a75519a clear cache remove index.html
zhangjing@preamble ~/P/fengerzh.github.io> git push -f
Total 0 (delta 0), reused 0 (delta 0)
To github.com:fengerzh/fengerzh.github.io.git
 + a3188cf...a75519a master -> master (forced update)
zhangjing@preamble ~/P/fengerzh.github.io> git pull
Already up-to-date.
zhangjing@preamble ~/P/fengerzh.github.io>
```

总结

以上4种状态的撤销我们都用到了同一个命令git reset --hard，前2种状态的用法甚至完全一样，所以只要掌握了git reset --hard这个命令的用法，从此你再也不用担心提交错误了。