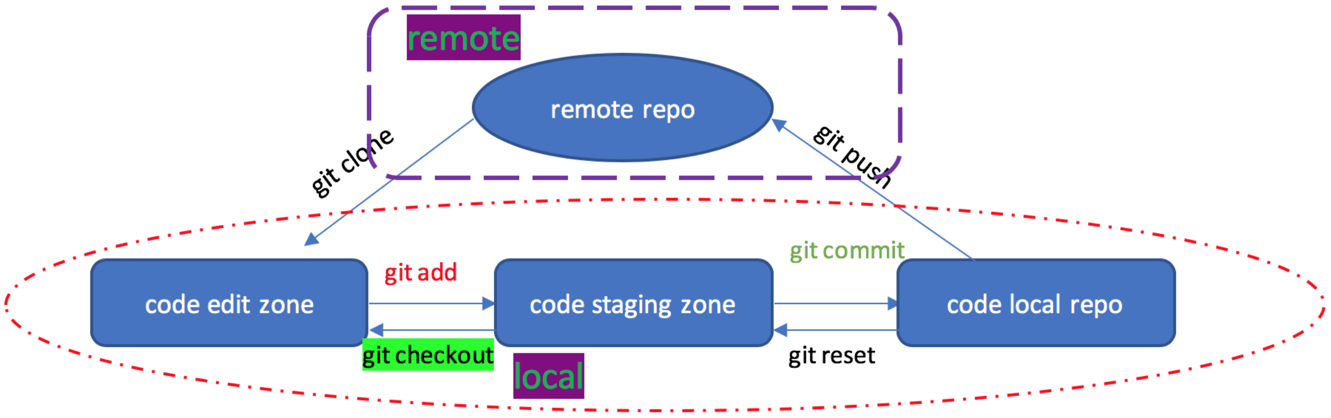
代码回退

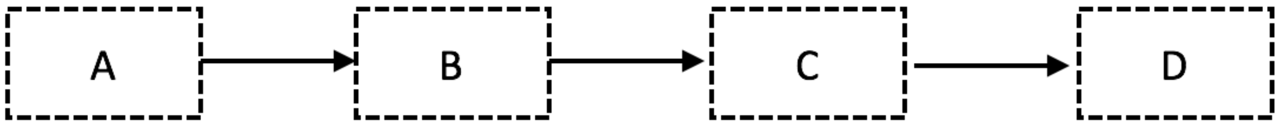
代码回退是一个所有程序员绕不开的话题，回退是指将代码从代码库回退到代码缓存区。git提供git reset和git revert两种方式来实现代码的回退。



1 git reset回退代码

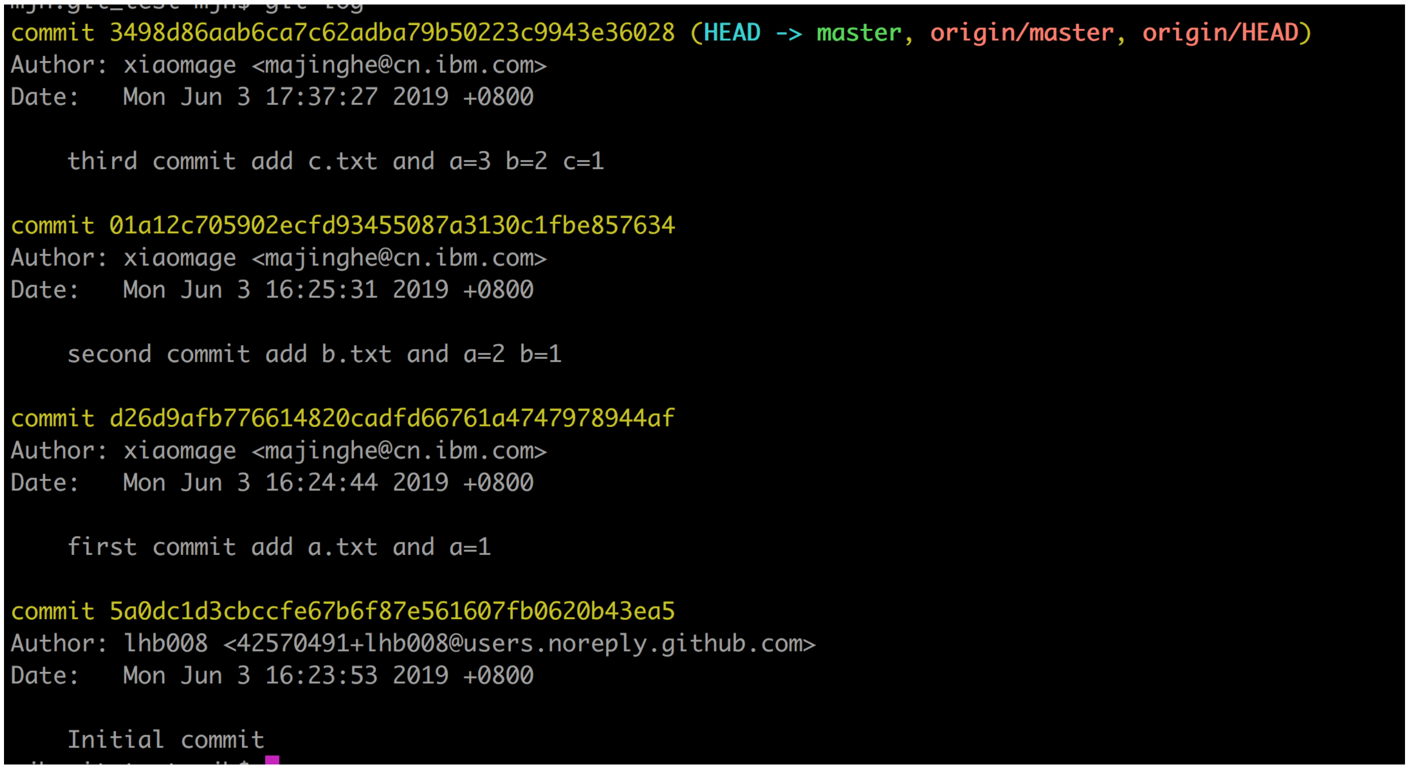
`git reset` 是将代码的当前分支的指针，重新指向了一个新的节点。新的节点就是代码想要被回退的节点。
`git reset`有三种模式`git reset --soft`，`git reset --mixed`，`git reset --hard`

比如说，某个git repo的提交历史如下，总共有四个commit节点，假如我们想要将代码回退至commit C，下面我们来分别看三种reset命令的作用。



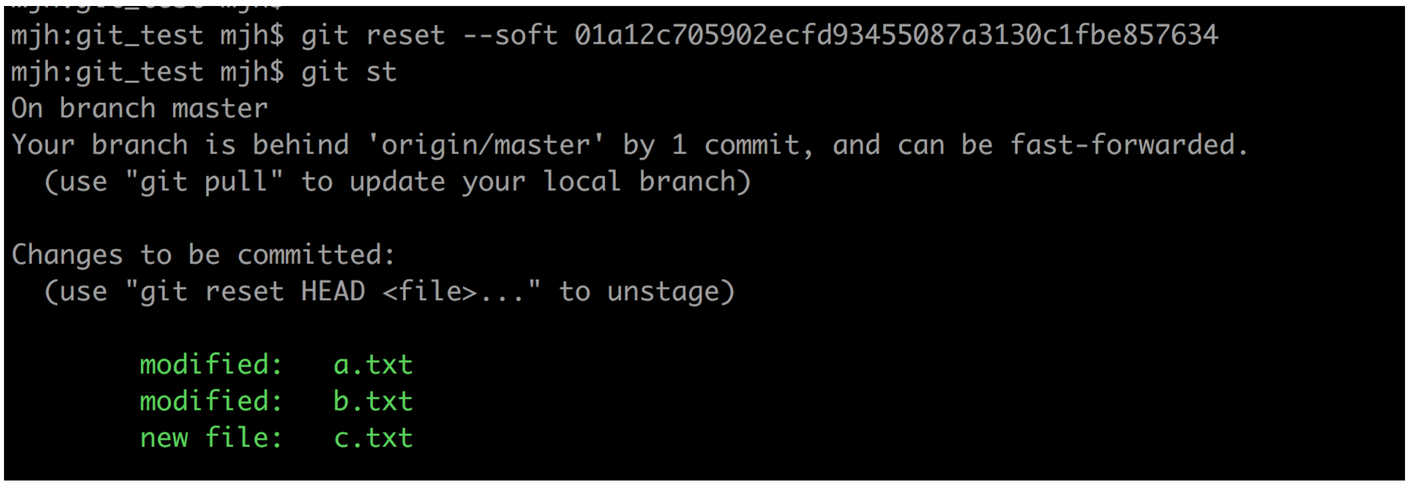
1.1 git reset –soft

`git reset --soft` 仅回退 HEAD，保留暂存区和工作目录中的内容不变。简单的说，只是回退了commit信息。



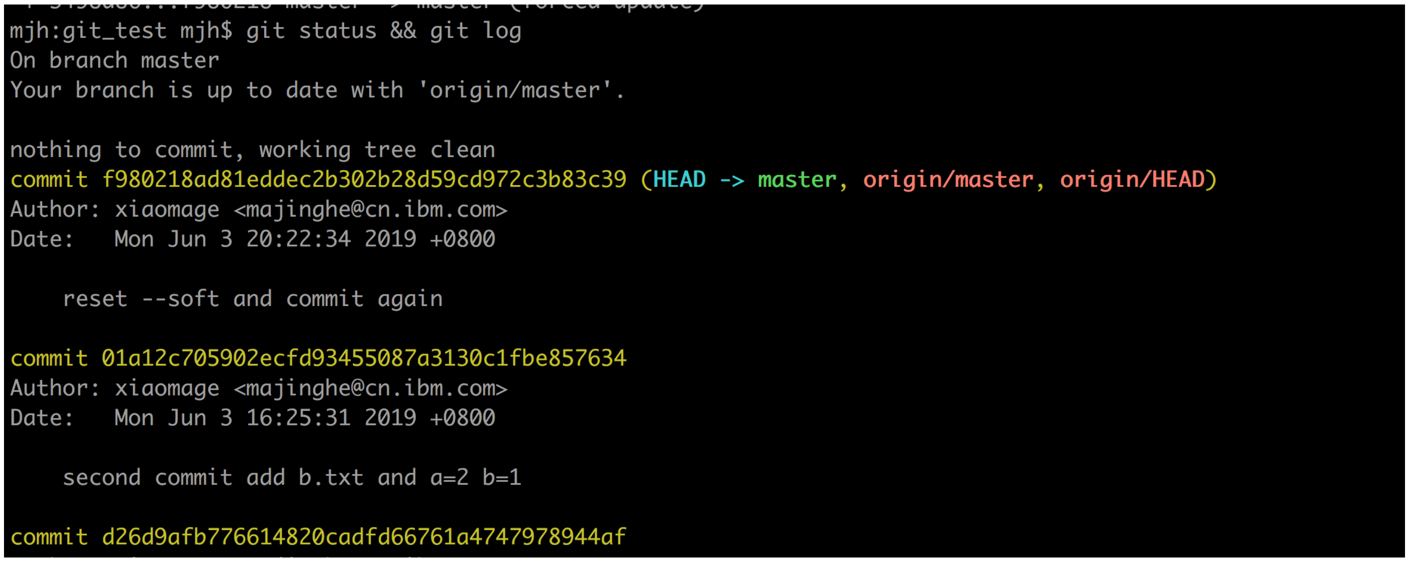
```
commit 3498d86aab6ca7c62adba79b50223c9943e36028 (HEAD -> master, origin/master, origin/HEAD)
Author: xiaomage <majinghe@cn.ibm.com>
Date:   Mon Jun 3 17:37:27 2019 +0800

    third commit add c.txt and a=3 b=2 c=1

commit 01a12c705902ecfd93455087a3130c1fbe857634
Author: xiaomage <majinghe@cn.ibm.com>
Date:   Mon Jun 3 16:25:31 2019 +0800

    second commit add b.txt and a=2 b=1

commit d26d9afb776614820cadfd66761a4747978944af
Author: xiaomage <majinghe@cn.ibm.com>
Date:   Mon Jun 3 16:24:44 2019 +0800

    first commit add a.txt and a=1

commit 5a0dc1d3cbccfe67b6f87e561607fb0620b43ea5
Author: lhb008 <42570491+lhb008@users.noreply.github.com>
Date:   Mon Jun 3 16:23:53 2019 +0800

    Initial commit
```

假如要将当前代码回退至commit C，也就是第二个提交节点，hash值是 `01a12c705902ecfd93455087a3130c1fbe857634` 。执行 `git reset --soft 01a12c705902ecfd93455087a3130c1fbe857634` ，然后用 `git status` 查看。



```
mjh:git_test mjh$ git reset --soft 01a12c705902ecfd93455087a3130c1fbe857634
mjh:git_test mjh$ git st
On branch master
Your branch is behind 'origin/master' by 1 commit, and can be fast-forwarded.
  (use "git pull" to update your local branch)

Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   a.txt
        modified:   b.txt
        new file:   c.txt
```

由上图可以看到，代码已经从本地仓库区回退值代码缓存区。由于只是回退了commit信息，如果要重新提交，可以重新用 `git commit` 命令进行重新提交。需要注意的是，重新commit之后，代码被提交到本地仓库，但是此时远端仓库(github)的代码依旧是在reset之前的。如果想用reset之后的代码对于远端库上面的代码进行覆盖(这种覆盖是极不推荐的，只是做demo讲述用，在实际工作中慎用。)可以执行 `git push -f` 进行代码推送。然后用 `git status && git log` 继续查看。

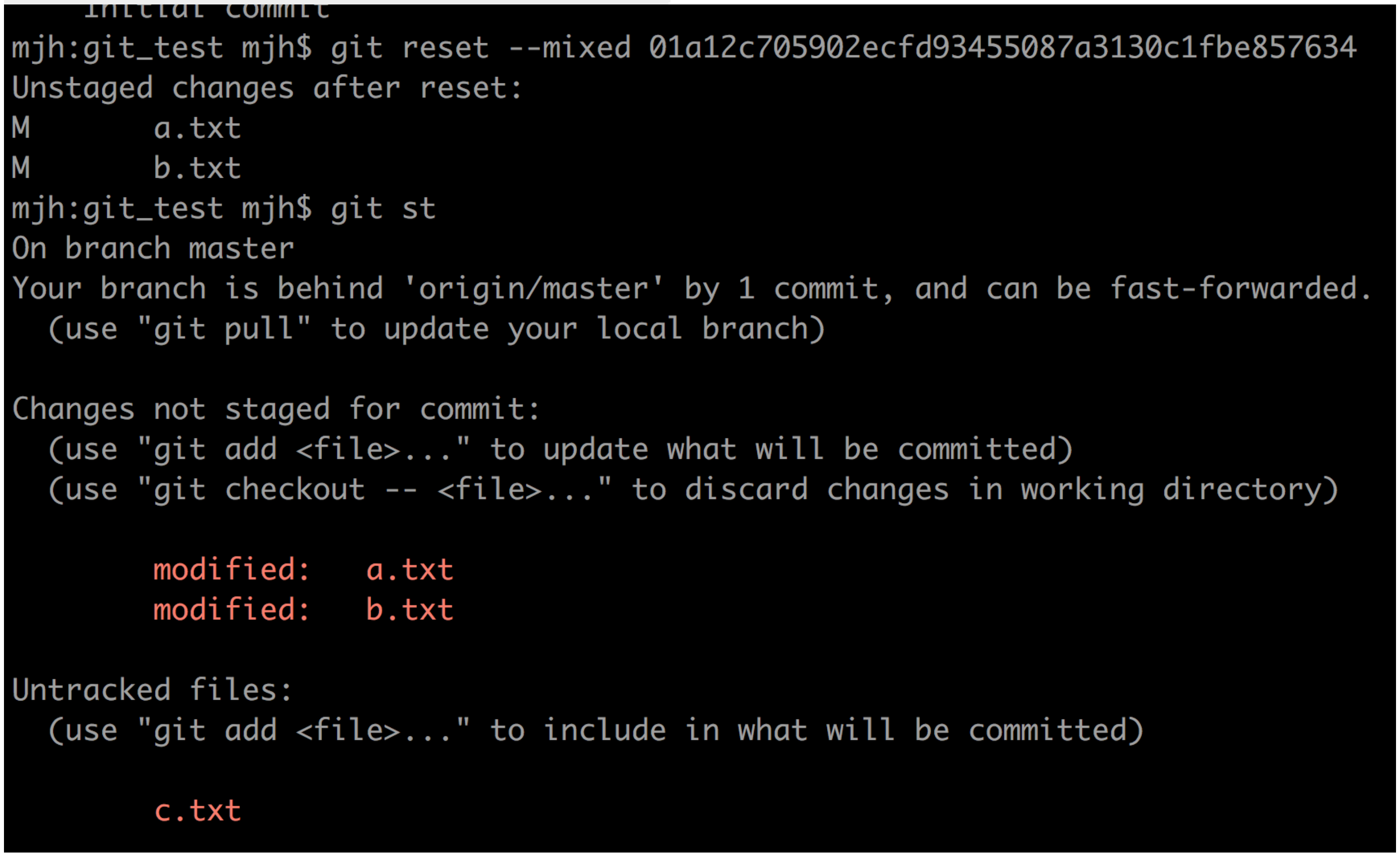


```
mjh:git_test mjh$ git status && git log
On branch master
Your branch is up to date with 'origin/master'.

nothing to commit, working tree clean
commit f980218ad81eddec2b302b28d59cd972c3b83c39 (HEAD -> master, origin/master, origin/HEAD)
Author: xiaomage <majinghe@cn.ibm.com>
Date:   Mon Jun 3 20:22:34 2019 +0800

    reset --soft and commit again

commit 01a12c705902ecfd93455087a3130c1fbe857634
Author: xiaomage <majinghe@cn.ibm.com>
Date:   Mon Jun 3 16:25:31 2019 +0800

    second commit add b.txt and a=2 b=1

commit d26d9afb776614820cadfd66761a4747978944af
```

由上图可以看出，代码已经被回退至C commit节点，然后被重新commit（`f980218ad81eddec2b302b28d59cd972c3b83c39`）然后提交至远端仓库。
由上述讲述可知，`git reset --soft` 使用的场景，主要是在提交代码时，commit信息书写有问题，想要对于commit信息进行回退，然后重新进行提交。

1.2 git reset —mixed

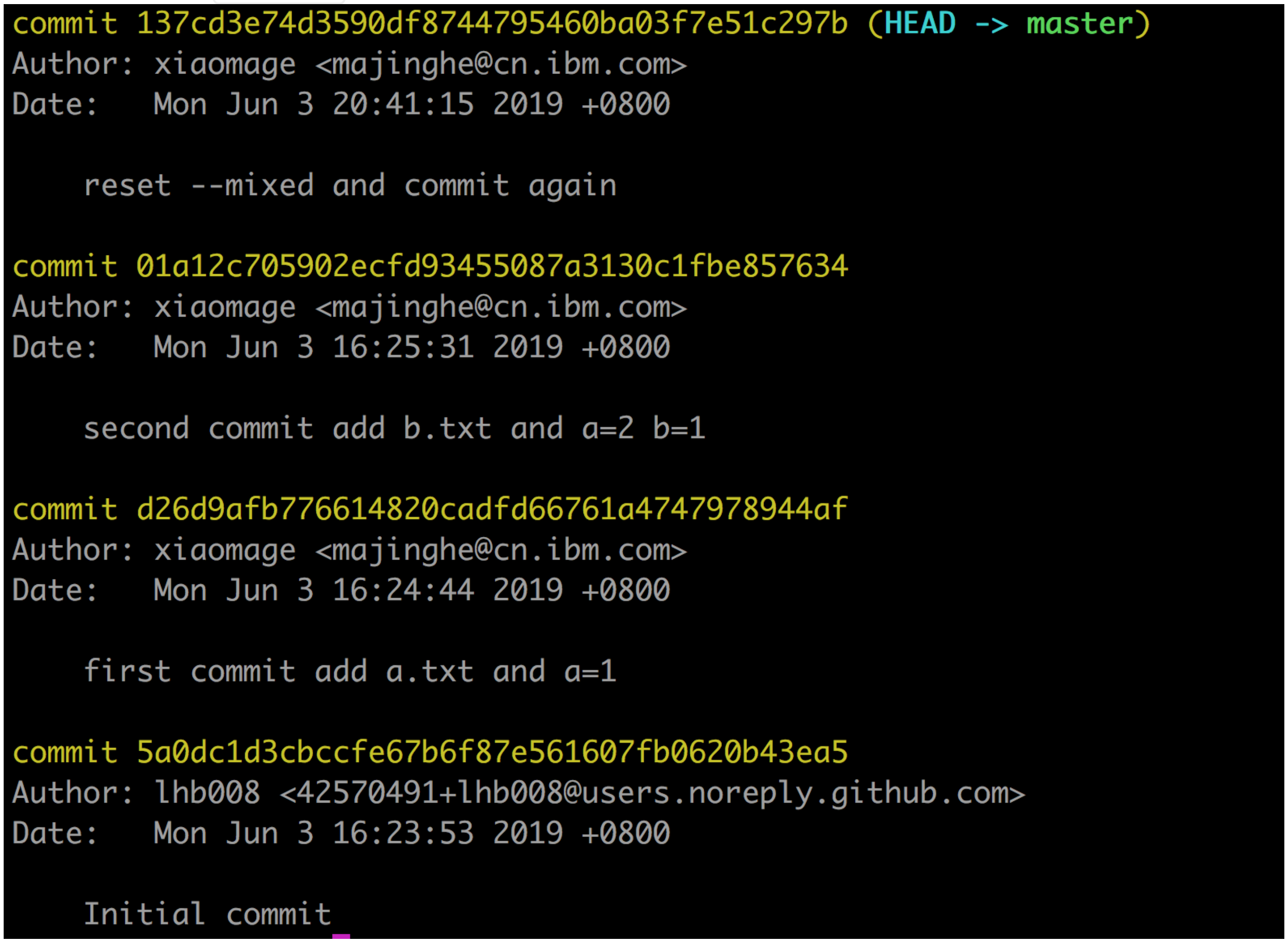
`git reset --mixed` 回退 HEAD 和 暂存区，保留工作目录中的内容不变。它是将代码直接从本地仓库回退至代码编辑区。在对上述例子，执行 `git reset --mixed 01a12c705902ecfd93455087a3130c1fbe857634` 命令后，显示如下：



```
mjh:git_test mjh$ git reset --mixed 01a12c705902ecfd93455087a3130c1fbe857634
Unstaged changes after reset:
M       a.txt
M       b.txt
mjh:git_test mjh$ git st
On branch master
Your branch is behind 'origin/master' by 1 commit, and can be fast-forwarded.
  (use "git pull" to update your local branch)

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   a.txt
        modified:   b.txt

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

        c.txt
```

如果需要重新提交，可以按照提交三步曲 `git add && git commit -m && git push` 将代码提交

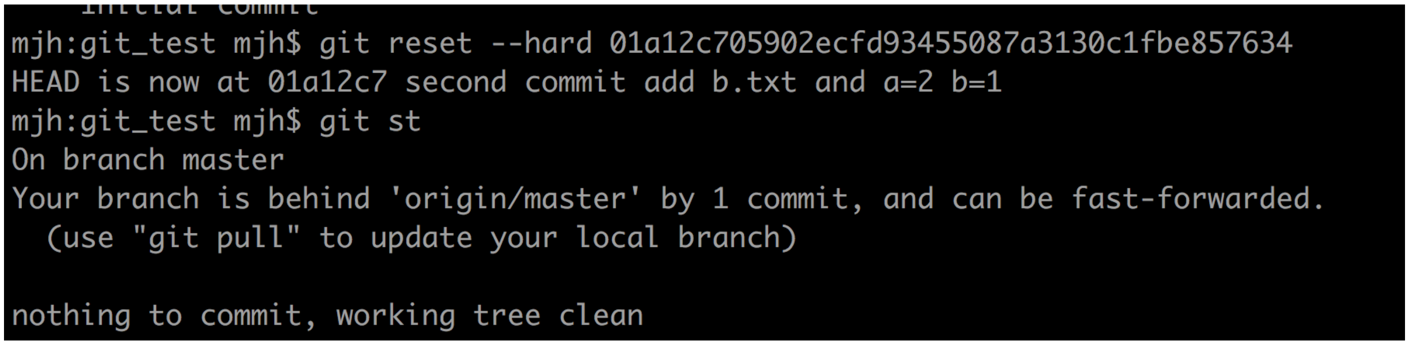
至远端仓库。用 `git log` 可以看到最新提交信息。



```
commit 137cd3e74d3590df8744795460ba03f7e51c297b (HEAD -> master)
Author: xiaomage <majinghe@cn.ibm.com>
Date:   Mon Jun 3 20:41:15 2019 +0800

    reset --mixed and commit again

commit 01a12c705902ecfd93455087a3130c1fbe857634
Author: xiaomage <majinghe@cn.ibm.com>
Date:   Mon Jun 3 16:25:31 2019 +0800

    second commit add b.txt and a=2 b=1

commit d26d9afb776614820cadfd66761a4747978944af
Author: xiaomage <majinghe@cn.ibm.com>
Date:   Mon Jun 3 16:24:44 2019 +0800

    first commit add a.txt and a=1

commit 5a0dc1d3cbccfe67b6f87e561607fb0620b43ea5
Author: lhb008 <42570491+lhb008@users.noreply.github.com>
Date:   Mon Jun 3 16:23:53 2019 +0800

    Initial commit
```

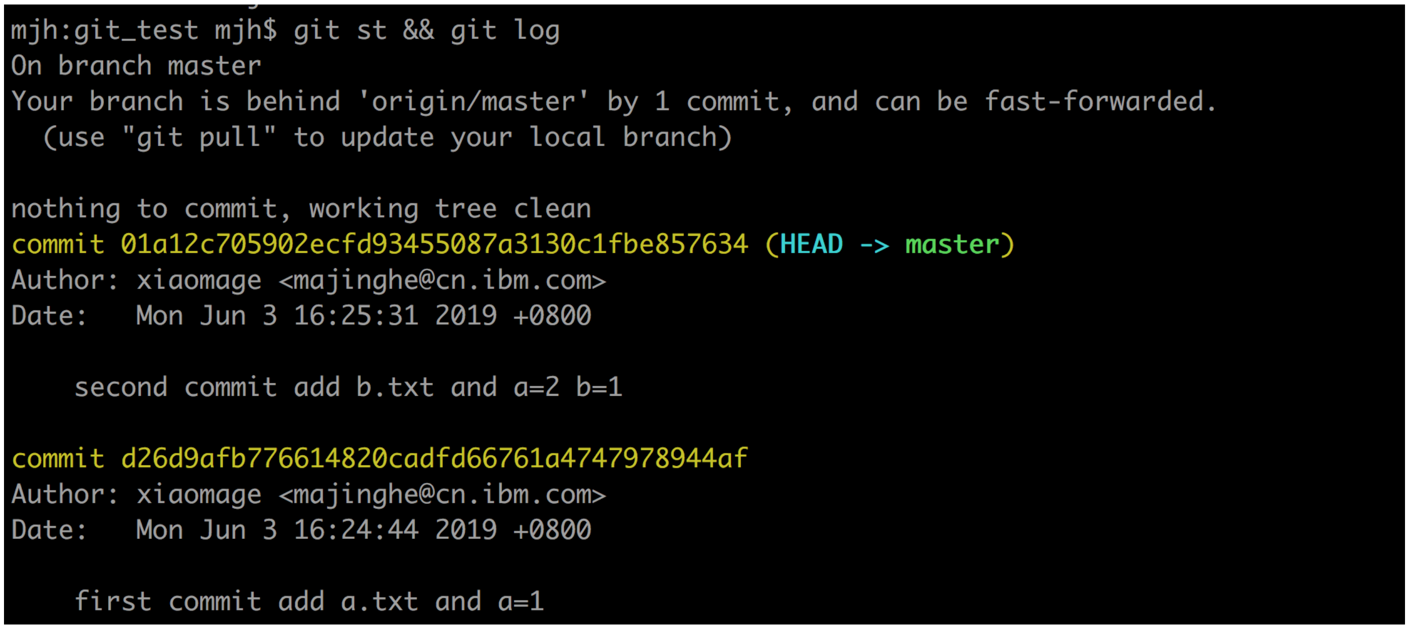
1.2 git reset —hard

`git reset --hard` 命令可以将代码直接从本地仓库区直接回退至代码编辑区之前。拿上面的例子来讲，如果用 `git reset --hard 01a12c705902ecfd93455087a3130c1fbe857634` 命令将代码进行回退。可以看到



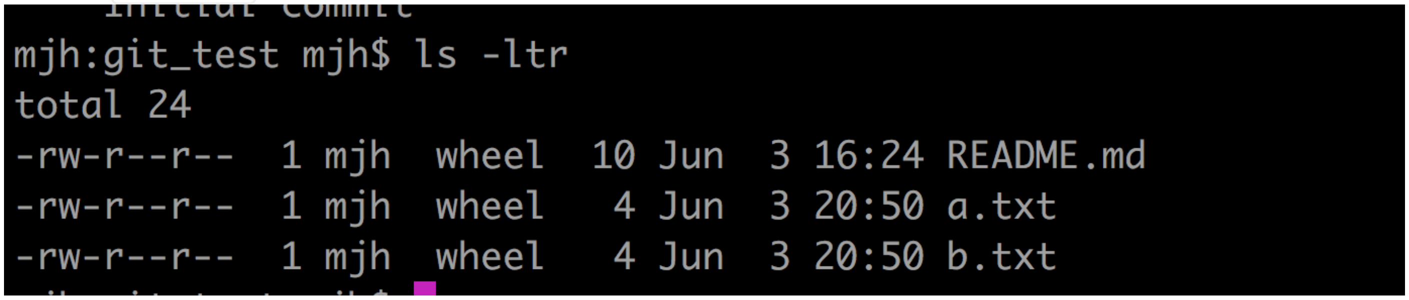
```
mjh:git_test mjh$ git reset --hard 01a12c705902ecfd93455087a3130c1fbe857634
HEAD is now at 01a12c7 second commit add b.txt and a=2 b=1
mjh:git_test mjh$ git st
On branch master
Your branch is behind 'origin/master' by 1 commit, and can be fast-forwarded.
  (use "git pull" to update your local branch)

nothing to commit, working tree clean
```

然后用 `git log` 查看提交记录



```
mjh:git_test mjh$ git st && git log
On branch master
Your branch is behind 'origin/master' by 1 commit, and can be fast-forwarded.
  (use "git pull" to update your local branch)

nothing to commit, working tree clean
commit 01a12c705902ecfd93455087a3130c1fbe857634 (HEAD -> master)
Author: xiaomage <majinghe@cn.ibm.com>
Date:   Mon Jun 3 16:25:31 2019 +0800

    second commit add b.txt and a=2 b=1

commit d26d9afb776614820cadfd66761a4747978944af
Author: xiaomage <majinghe@cn.ibm.com>
Date:   Mon Jun 3 16:24:44 2019 +0800

    first commit add a.txt and a=1
```

,如果用 `ls -ltr` ,查看文件目录，可以看到

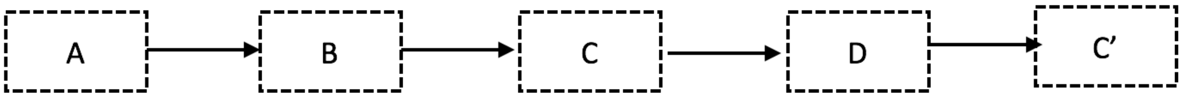


```
    initial commit
mjh:git_test mjh$ ls -ltr
total 24
-rw-r--r--  1 mjh  wheel  10 Jun  3 16:24 README.md
-rw-r--r--  1 mjh  wheel   4 Jun  3 20:50 a.txt
-rw-r--r--  1 mjh  wheel   4 Jun  3 20:50 b.txt
```

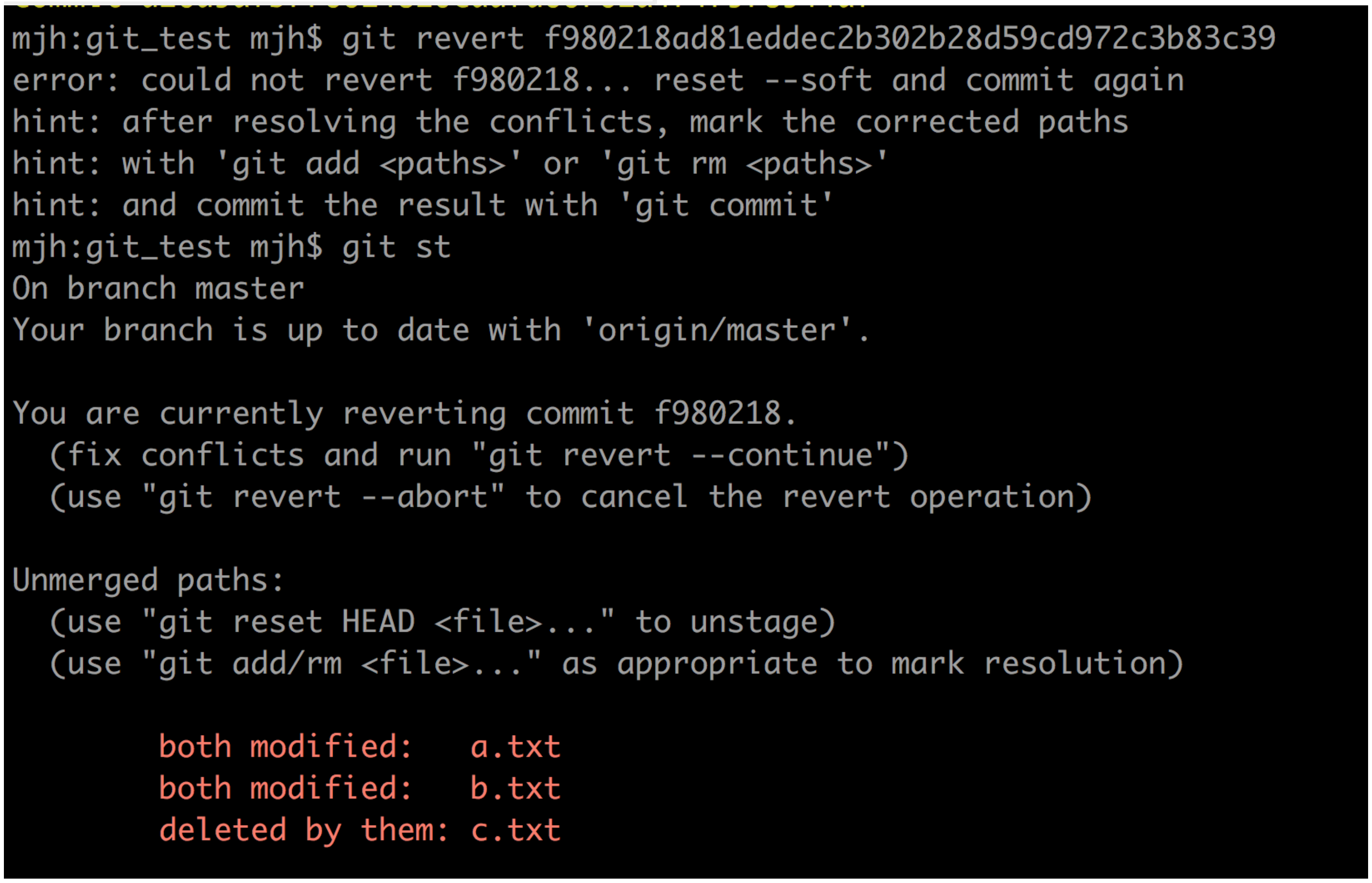
,文件c.txt文件不存在了。就证明了如果用 `git reset --hard` 命令，是将代码回退至代码编辑区之前，也就是说在将c.txt文件添加之前。
从上面可以看到。 `git reset` 命令回退代码，比较适合只是回退某一单的commit信息，或者想一次回退多单，并将回退后的代码对于远端库进行覆盖的场景。而且可以看到，当回退到commit C之后，再次进行提交时，commit D的提交历史将不会存在于提交历史中的。而这正是 `git resete` 与 `git revert` 最大的区别。

2 git revert回退代码

`git revert` 也是回退某次提交，但是回退节点前和后的commit信息以及提交历史都不会丢失，而且会将此次撤销操作作为一个新的提交，从而形成一个新的commit节点。假如将代码用 `git revert` 命令回退至commit C，则最终的commit节点和提交历史，如下图

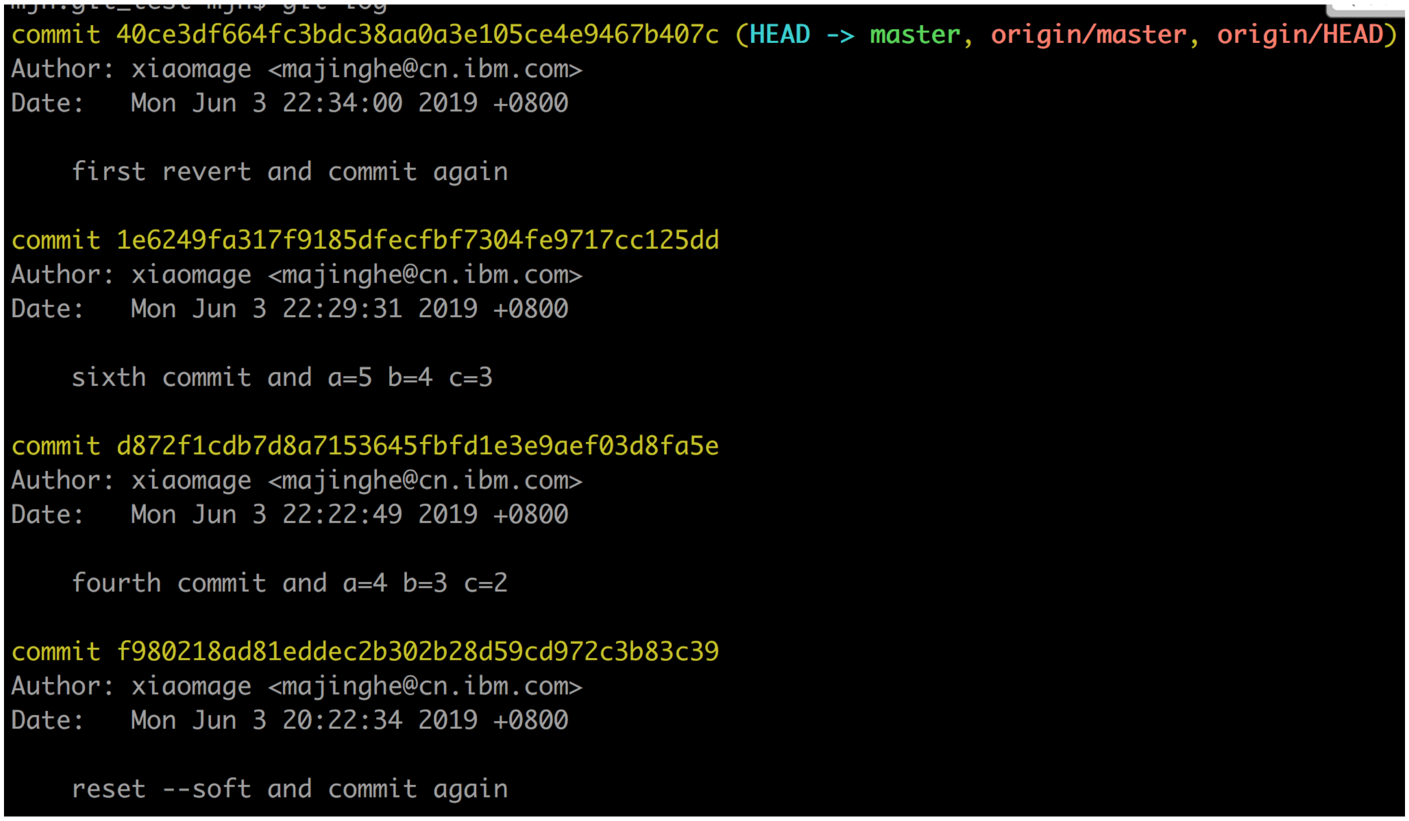


首先用 `git log` 查看提交记录，然后找到需要回退的commit id(比如 `f980218ad81eddec2b302b28d59cd972c3b83c39`)然后执行 `git revert` `f980218ad81eddec2b302b28d59cd972c3b83c39` ，可以看到如下信息：



```
mjh:git_test mjh$ git revert f980218ad81eddec2b302b28d59cd972c3b83c39
error: could not revert f980218... reset --soft and commit again
hint: after resolving the conflicts, mark the corrected paths
hint: with 'git add <paths>' or 'git rm <paths>'
hint: and commit the result with 'git commit'
mjh:git_test mjh$ git st
On branch master
Your branch is up to date with 'origin/master'.

You are currently reverting commit f980218.
  (fix conflicts and run "git revert --continue")
  (use "git revert --abort" to cancel the revert operation)

Unmerged paths:
  (use "git reset HEAD <file>..." to unstage)
  (use "git add/rm <file>..." as appropriate to mark resolution)

        both modified:   a.txt
        both modified:   b.txt
        deleted by them: c.txt
```

。回退过程中有可能会有冲突，如果有按照提示解决冲突之后，重新commit，然后再用git log查看提交记录



```
commit 40ce3df664fc3bdc38aa0a3e105ce4e9467b407c (HEAD -> master, origin/master, origin/HEAD)
Author: xiaomage <majinghe@cn.ibm.com>
Date:   Mon Jun 3 22:34:00 2019 +0800

    first revert and commit again

commit 1e6249fa317f9185dfecfbf7304fe9717cc125dd
Author: xiaomage <majinghe@cn.ibm.com>
Date:   Mon Jun 3 22:29:31 2019 +0800

    sixth commit and a=5 b=4 c=3

commit d872f1cdb7d8a7153645fbfd1e3e9aef03d8fa5e
Author: xiaomage <majinghe@cn.ibm.com>
Date:   Mon Jun 3 22:22:49 2019 +0800

    fourth commit and a=4 b=3 c=2

commit f980218ad81eddec2b302b28d59cd972c3b83c39
Author: xiaomage <majinghe@cn.ibm.com>
Date:   Mon Jun 3 20:22:34 2019 +0800

    reset --soft and commit again
```

,可以清晰的看到回退节点 `f980218ad81eddec2b302b28d59cd972c3b83c39` 之后的提交记

录 `d872f1cdb7d8a7153645fbfd1e3e9aef03d8fa5e` 和 `1e6249fa317f9185dfecfbf7304fe9717cc125dd` 都存在，并没有因为revert而被撤销，而且新增了一个节点 `40ce3df664fc3bdc38aa0a3e105ce4e9467b407c` ,这就是 `git revert` 再执行代码回退时候，把整个回退操作当作一个commit，进行再次提交，从而进行记录。