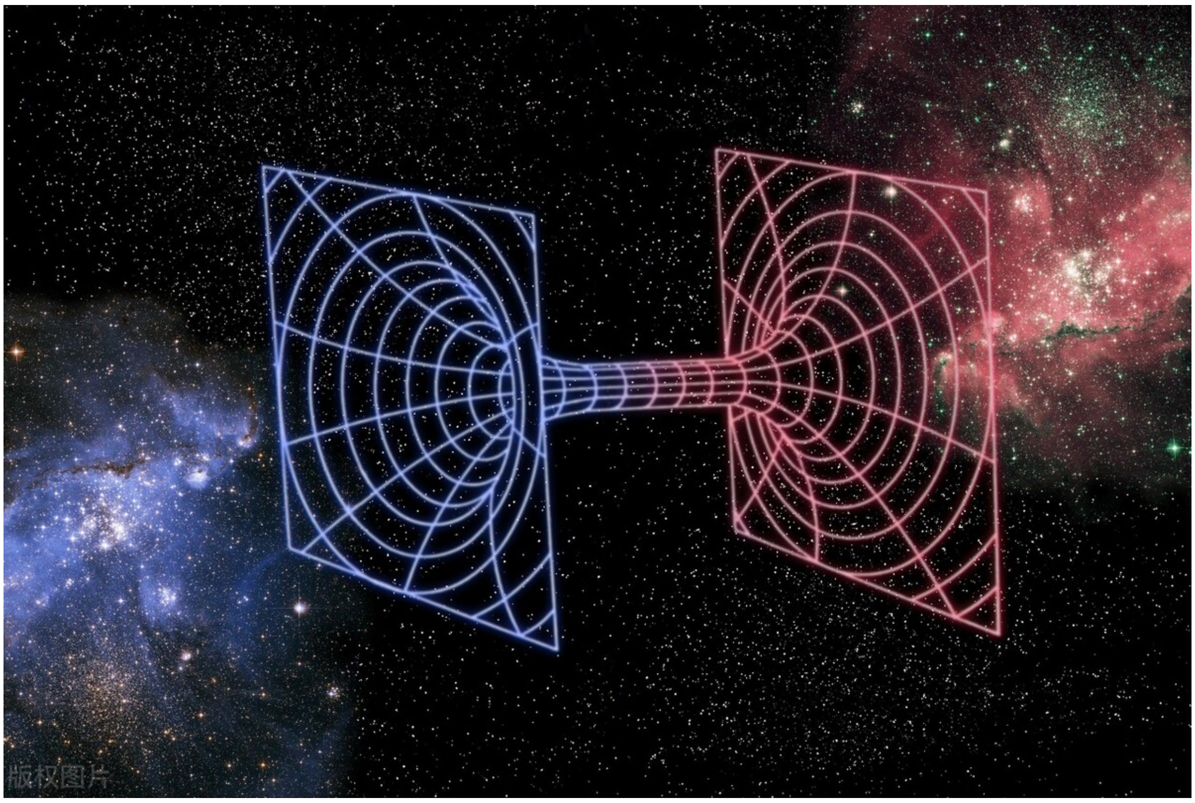
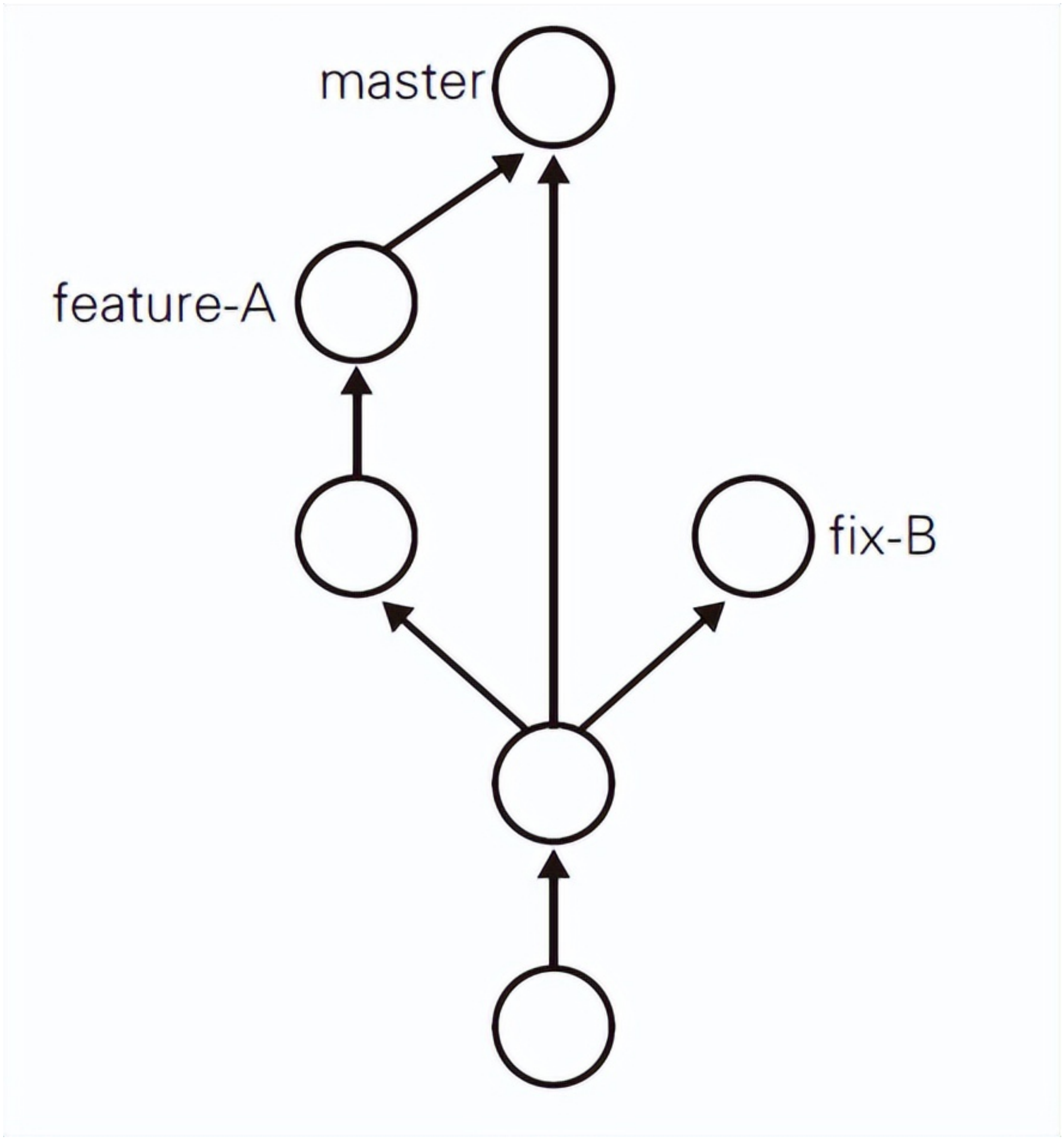
什么是分支?

分支可以想象成电影中的平行宇宙，或者说你有多个分身，假定你要实现一个视频教学网站项目，平行宇宙中的你同时进行后端逻辑处理、数据库管理、前端页面设计、美工设计、运维和安全管理，**最终所有分支项目合并到一个主分支，就形成了完整项目。**

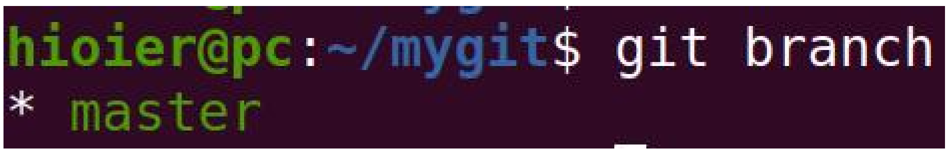


实际开发过程中分支有什么作用呢？比如，你要实现一个功能模块，但是需要两周时间，不完整的代码不能运行，为了不干扰其他人的正常使用，你下载当前的源码，创建一个分支开始开发，两周之后再合并到主分支中。（master是默认的主分支）



查看和创建分支

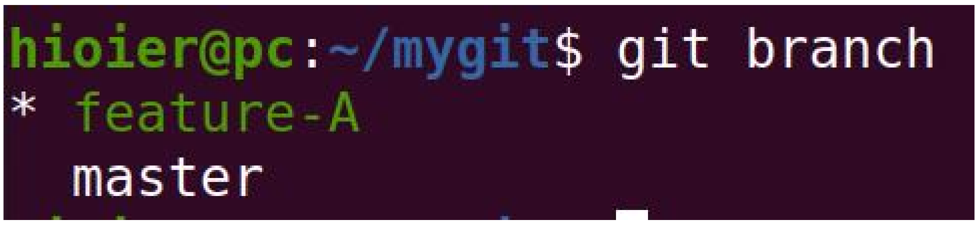
查看当前分支：`git branch`



```
hioier@pc:~/mygit$ git branch
* master
```

创建分支：`git branch feature-A`

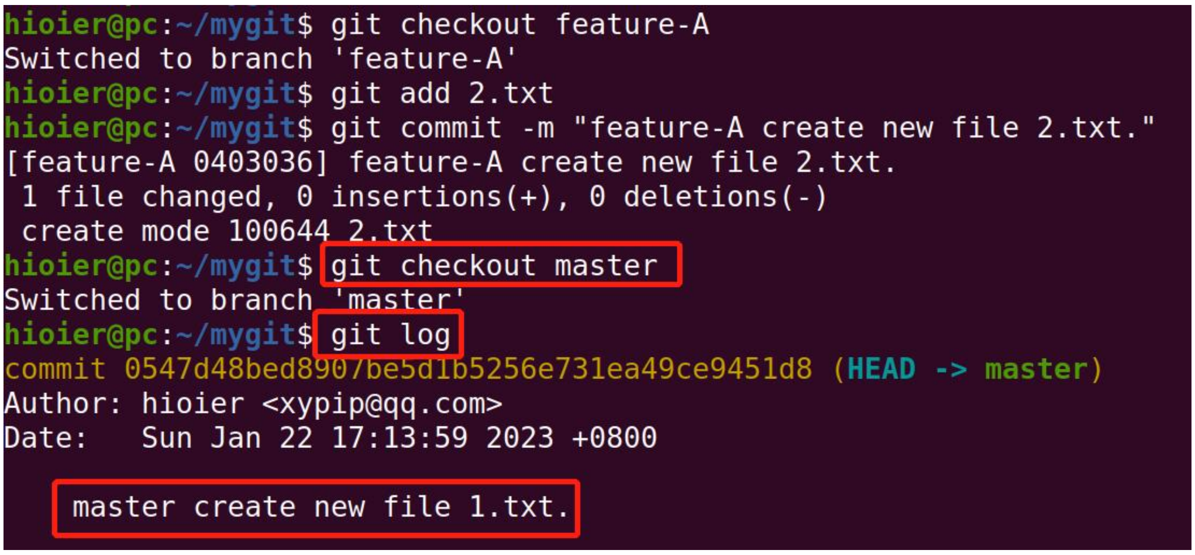
切换分支：`git checkout feature-A`



```
hioier@pc:~/mygit$ git branch
* feature-A
  master
```

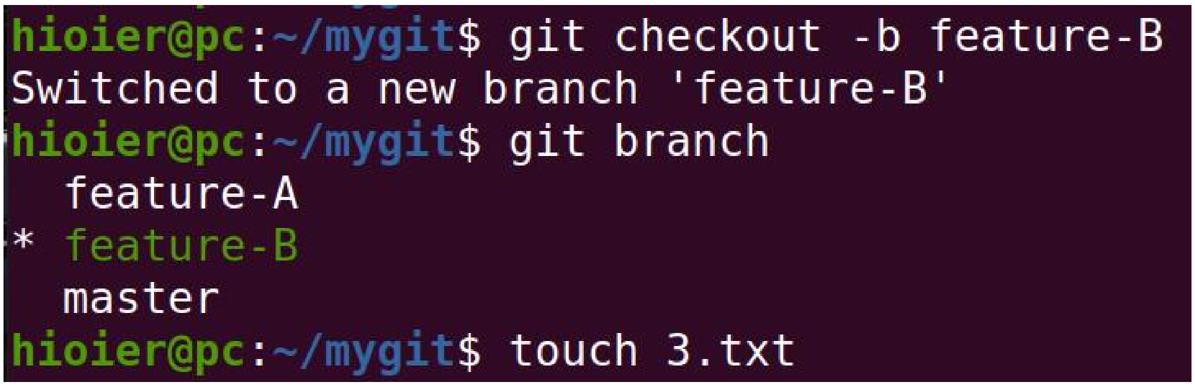
“*”表示我们当前所在的分支。

在feature-A分支commit 创建2.txt版本，切换回master分支，发现历史版本记录还是master分支commit的1.txt版本：



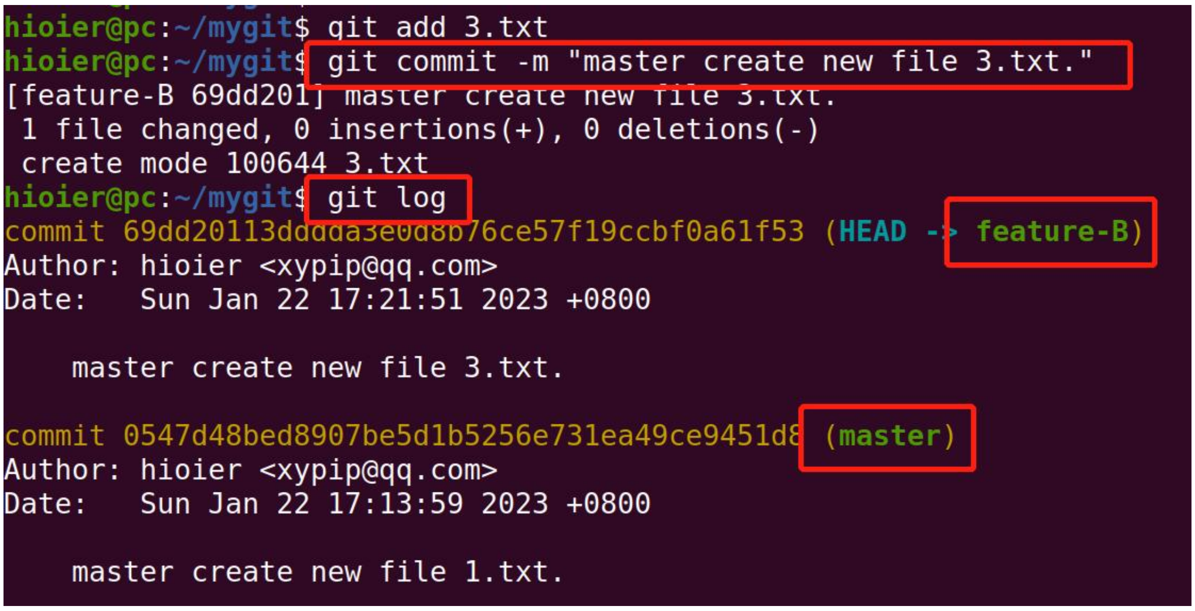
```
hioier@pc:~/mygit$ git checkout feature-A
Switched to branch 'feature-A'
hioier@pc:~/mygit$ git add 2.txt
hioier@pc:~/mygit$ git commit -m "feature-A create new file 2.txt."
[feature-A 0403036] feature-A create new file 2.txt.
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 2.txt
hioier@pc:~/mygit$ git checkout master
Switched to branch 'master'
hioier@pc:~/mygit$ git log
commit 0547d48bed8907be5d1b5256e731ea49ce9451d8 (HEAD -> master)
Author: hioier <xypip@qq.com>
Date:   Sun Jan 22 17:13:59 2023 +0800

    master create new file 1.txt.
```

创建和切换分支合并：



```
hioier@pc:~/mygit$ git checkout -b feature-B
Switched to a new branch 'feature-B'
hioier@pc:~/mygit$ git branch
  feature-A
* feature-B
  master
hioier@pc:~/mygit$ touch 3.txt
```

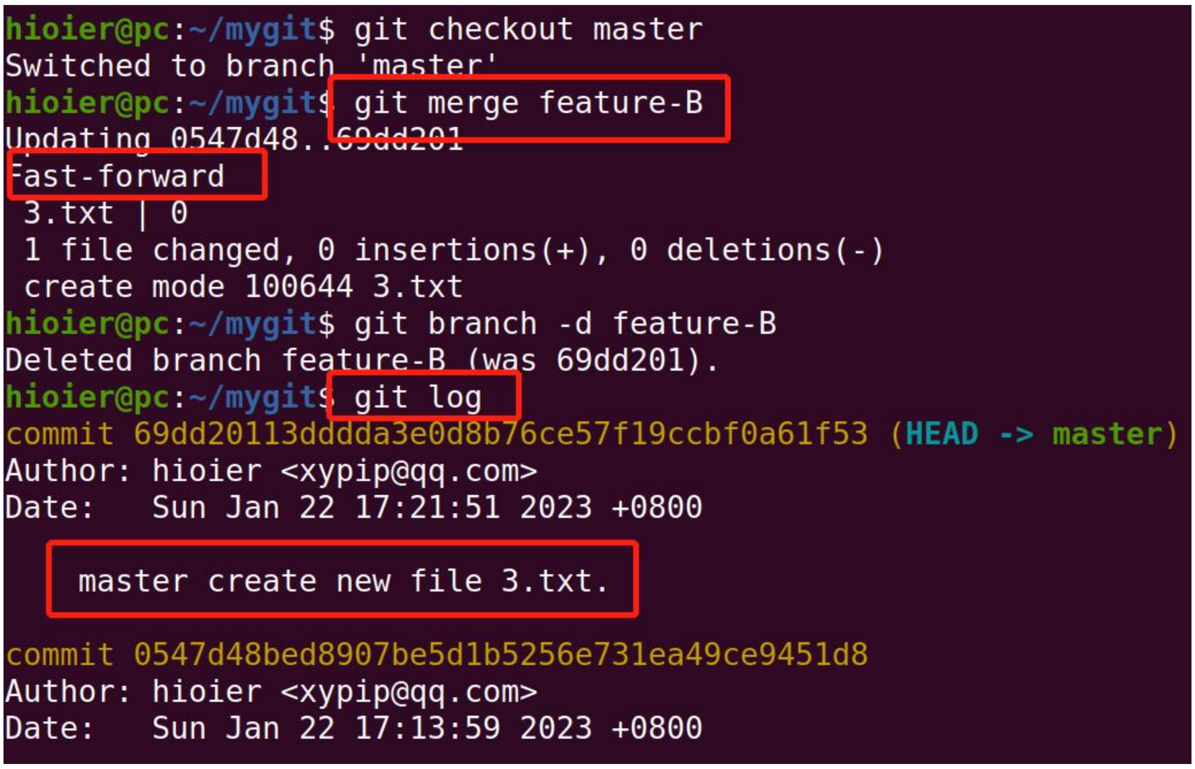
然后commit：



```
hioier@pc:~/mygit$ git add 3.txt
hioier@pc:~/mygit$ git commit -m "master create new file 3.txt."
[feature-B 69dd201] master create new file 3.txt.
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 3.txt
hioier@pc:~/mygit$ git log
commit 69dd20113dddda3e0d8b76ce57f19ccbf0a61f53 (HEAD -> feature-B)
Author: hioier <xypip@qq.com>
Date:   Sun Jan 22 17:21:51 2023 +0800

    master create new file 3.txt.

commit 0547d48bed8907be5d1b5256e731ea49ce9451d8 (master)
Author: hioier <xypip@qq.com>
Date:   Sun Jan 22 17:13:59 2023 +0800

    master create new file 1.txt.
```

合并分支

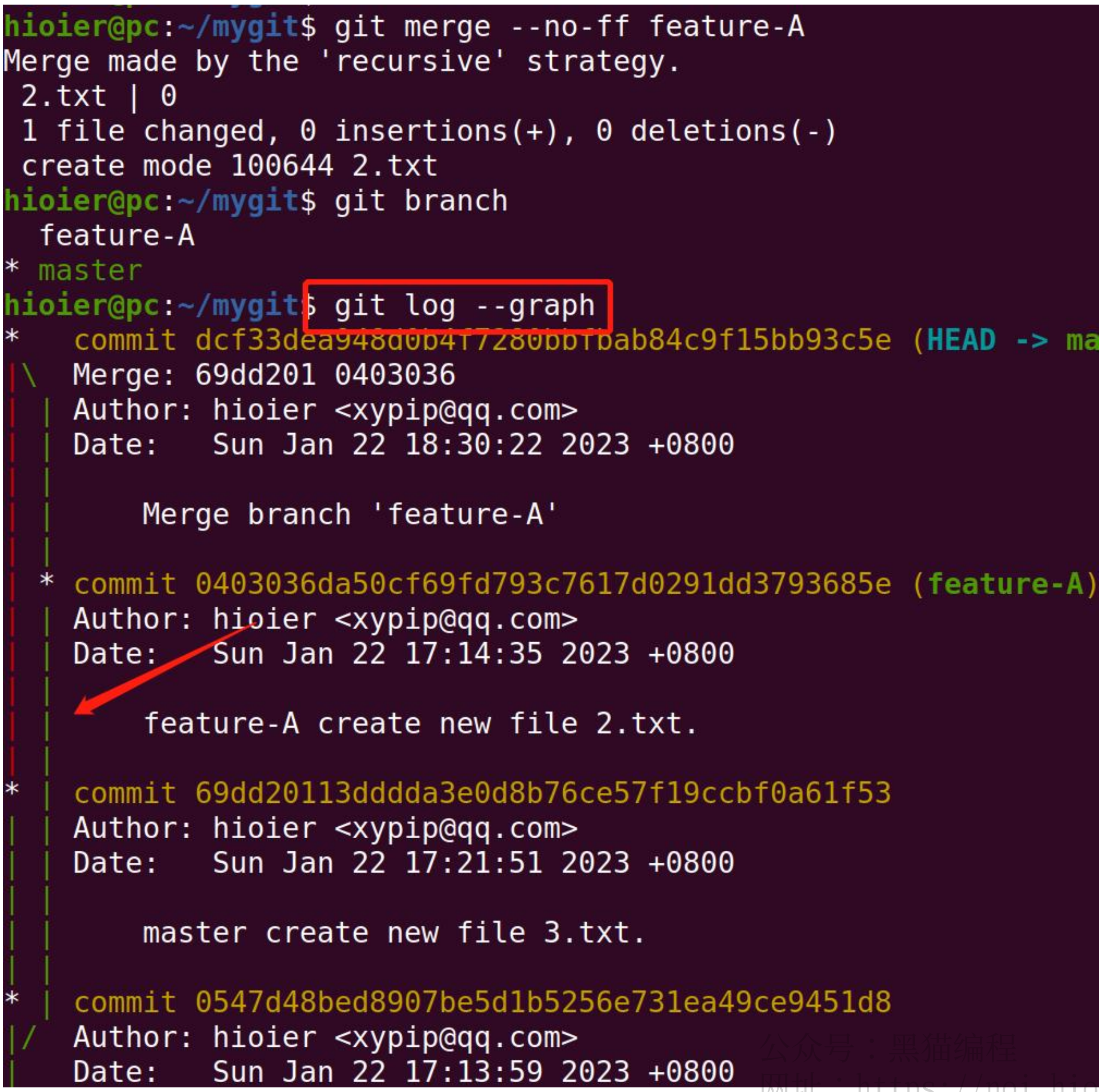
将feature-B合并到master上，然后删除feature-B分支：



```
hioier@pc:~/mygit$ git checkout master
Switched to branch 'master'
hioier@pc:~/mygit$ git merge feature-B
Updating 0547d48..69dd201
Fast-forward
 3.txt | 0
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 3.txt
hioier@pc:~/mygit$ git branch -d feature-B
Deleted branch feature-B (was 69dd201).
hioier@pc:~/mygit$ git log
commit 69dd20113dddda3e0d8b76ce57f19ccbf0a61f53 (HEAD -> master)
Author: hioier <xypip@qq.com>
Date:   Sun Jan 22 17:21:51 2023 +0800

    master create new file 3.txt.

commit 0547d48bed8907be5d1b5256e731ea49ce9451d8
Author: hioier <xypip@qq.com>
Date:   Sun Jan 22 17:13:59 2023 +0800
```

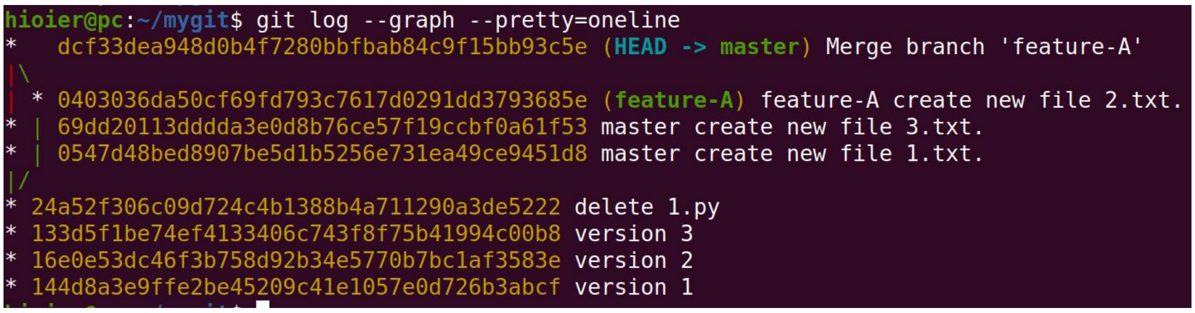
这是一种快速合并方法，直接将HEAD指针指向feature-B。

再合并feature-A分支到master。取消快速合并：`git merge --no-ff feature-A`



```
hioier@pc:~/mygit$ git merge --no-ff feature-A
Merge made by the 'recursive' strategy.
 2.txt | 0
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 2.txt
hioier@pc:~/mygit$ git branch
  feature-A
* master
hioier@pc:~/mygit$ git log --graph
*   commit dcf33dea948d0b4f7280bbfbab84c9f15bb93c5e (HEAD -> ma
|\  Merge: 69dd201 0403036
| | Author: hioier <xypip@qq.com>
| | Date:   Sun Jan 22 18:30:22 2023 +0800
| |
| |     Merge branch 'feature-A'
| |
| * commit 0403036da50cf69fd793c7617d0291dd3793685e (feature-A)
| | Author: hioier <xypip@qq.com>
| | Date:   Sun Jan 22 17:14:35 2023 +0800
| |
| |     feature-A create new file 2.txt.
| |
* | commit 69dd20113dddda3e0d8b76ce57f19ccbf0a61f53
| | Author: hioier <xypip@qq.com>
| | Date:   Sun Jan 22 17:21:51 2023 +0800
| |
| |     master create new file 3.txt.
| |
* | commit 0547d48bed8907be5d1b5256e731ea49ce9451d8
|/  Author: hioier <xypip@qq.com>
|   Date:   Sun Jan 22 17:13:59 2023 +0800
```

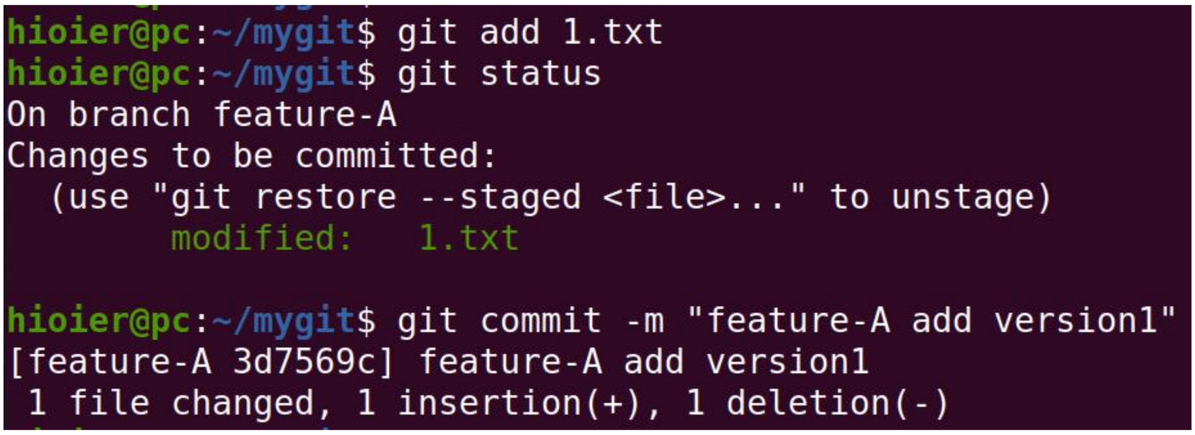
以简短形式查看：



```
hioier@pc:~/mygit$ git log --graph --pretty=oneline
*   dcf33dea948d0b4f7280bbfbab84c9f15bb93c5e (HEAD -> master) Merge branch 'feature-A'
|\
| * 0403036da50cf69fd793c7617d0291dd3793685e (feature-A) feature-A create new file 2.txt.
* | 69dd20113dddda3e0d8b76ce57f19ccbf0a61f53 master create new file 3.txt.
* | 0547d48bed8907be5d1b5256e731ea49ce9451d8 master create new file 1.txt.
|/
* 24a52f306c09d724c4b1388b4a711290a3de5222 delete 1.py
* 133d5f1be74ef4133406c743f8f75b41994c00b8 version 3
* 16e0e53dc46f3b758d92b34e5770b7bc1af3583e version 2
* 144d8a3e9ffe2be45209c41e1057e0d726b3abcf version 1
```

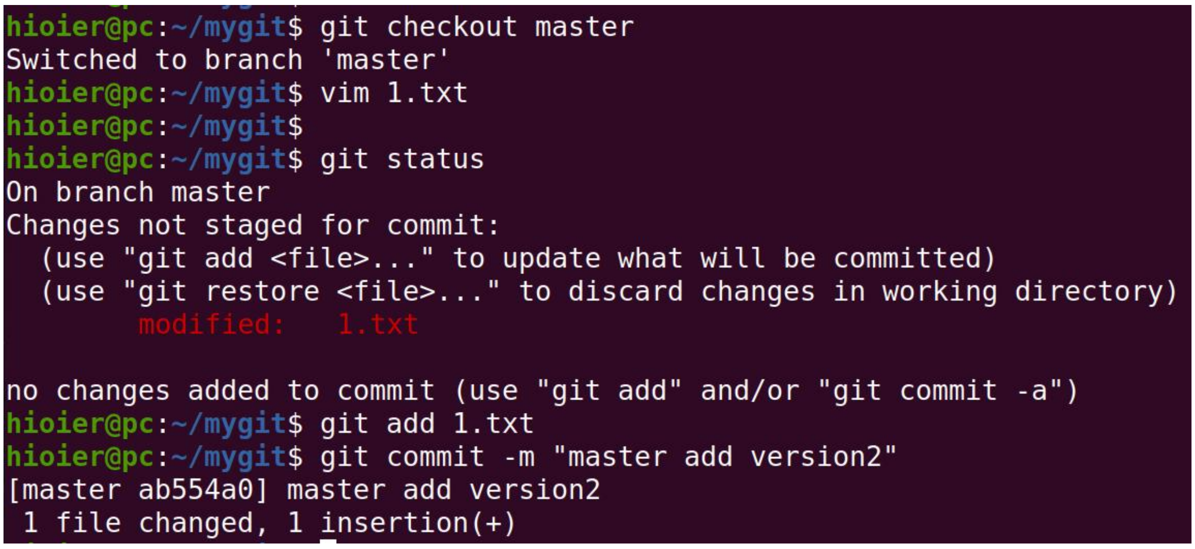
解决冲突

切换到A分支，修改1.txt，并commit：



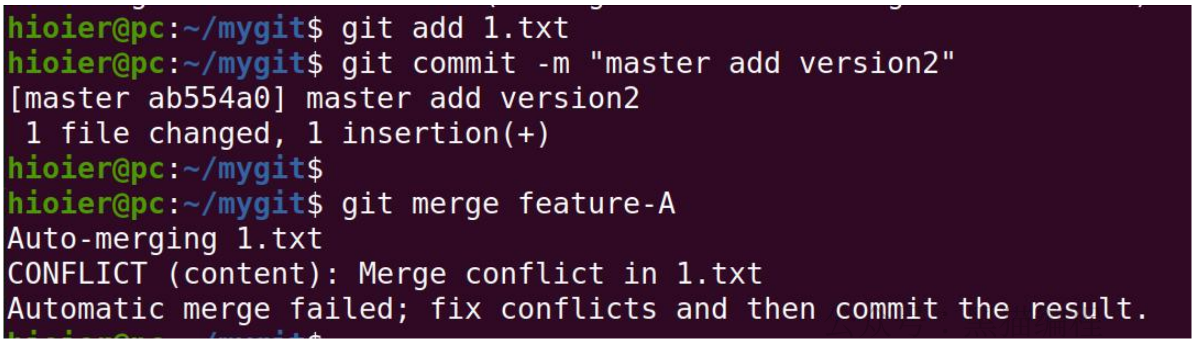
```
hioier@pc:~/mygit$ git add 1.txt
hioier@pc:~/mygit$ git status
On branch feature-A
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   1.txt

hioier@pc:~/mygit$ git commit -m "feature-A add version1"
[feature-A 3d7569c] feature-A add version1
 1 file changed, 1 insertion(+), 1 deletion(-)
```

切换到master分支，修改1.txt，并commit：



```
hioier@pc:~/mygit$ git checkout master
Switched to branch 'master'
hioier@pc:~/mygit$ vim 1.txt
hioier@pc:~/mygit$
hioier@pc:~/mygit$ 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:   1.txt

no changes added to commit (use "git add" and/or "git commit -a")
hioier@pc:~/mygit$ git add 1.txt
hioier@pc:~/mygit$ git commit -m "master add version2"
[master ab554a0] master add version2
 1 file changed, 1 insertion(+)
```

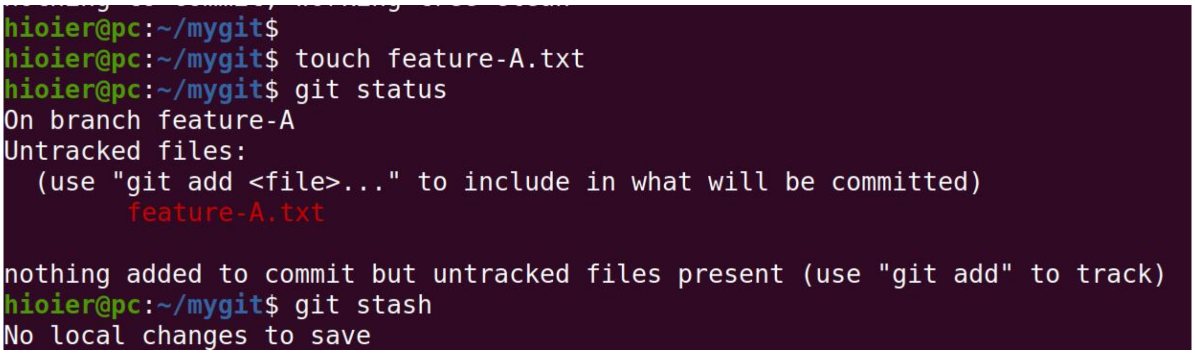
合并后，会产生冲突，要进行手动合并：



```
hioier@pc:~/mygit$ git add 1.txt
hioier@pc:~/mygit$ git commit -m "master add version2"
[master ab554a0] master add version2
 1 file changed, 1 insertion(+)
hioier@pc:~/mygit$
hioier@pc:~/mygit$ git merge feature-A
Auto-merging 1.txt
CONFLICT (content): Merge conflict in 1.txt
Automatic merge failed; fix conflicts and then commit the result.
```

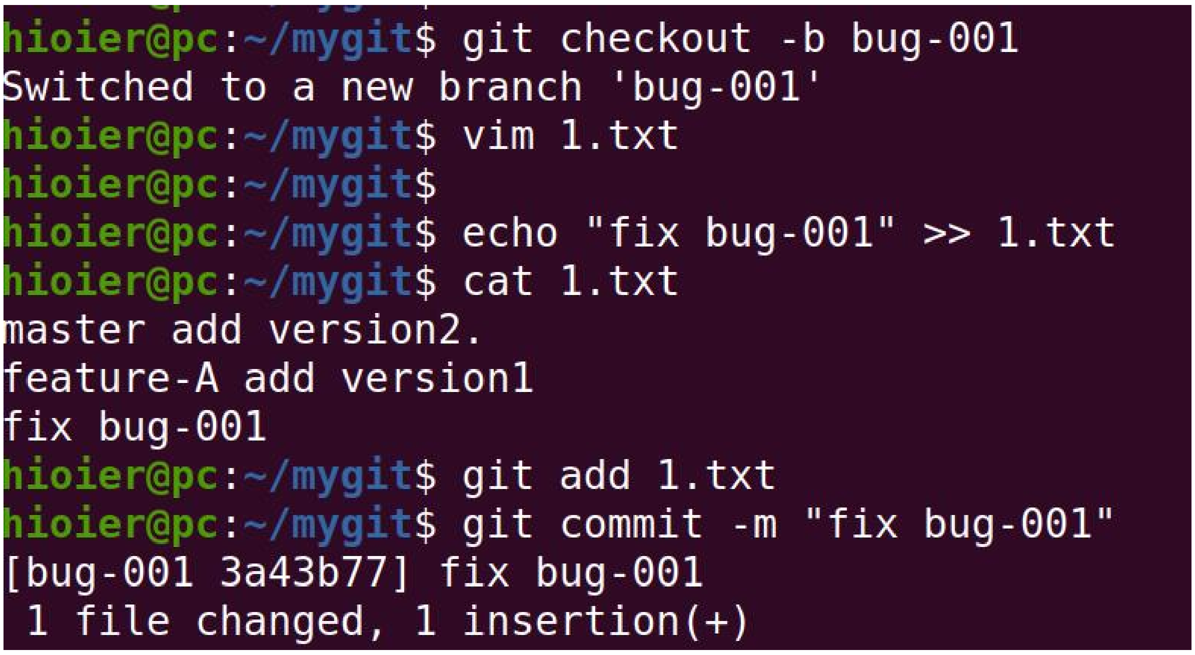
bug分支

当前正在feature-A工作，还未add到暂存区，突然有一个紧急的bug要进行修复：

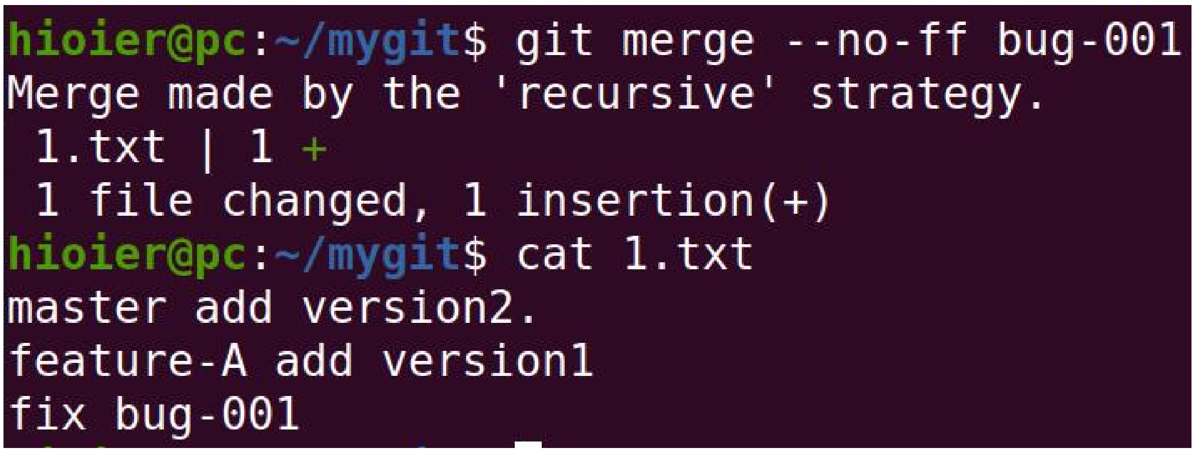


```
hioier@pc:~/mygit$
hioier@pc:~/mygit$ touch feature-A.txt
hioier@pc:~/mygit$ git status
On branch feature-A
Untracked files:
  (use "git add <file>..." to include in what will be committed)
	feature-A.txt

nothing added to commit but untracked files present (use "git add" to track)
hioier@pc:~/mygit$ git stash
No local changes to save
```

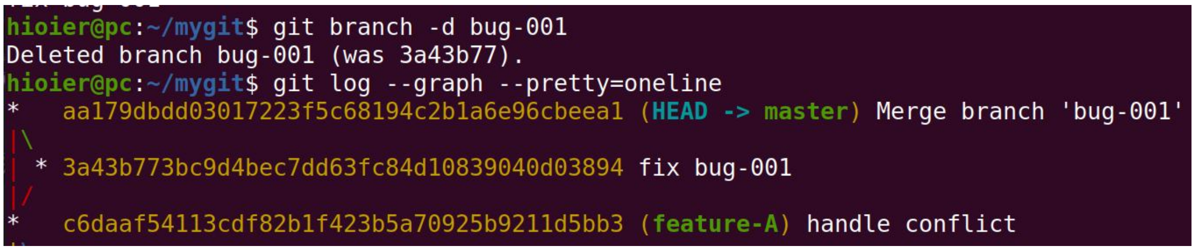
我们先将当前现场保存起来，创建bug分支进行修复，然后切回master，合并，删除bug分支：



```
hioier@pc:~/mygit$ git checkout -b bug-001
Switched to a new branch 'bug-001'
hioier@pc:~/mygit$ vim 1.txt
hioier@pc:~/mygit$
hioier@pc:~/mygit$ echo "fix bug-001" >> 1.txt
hioier@pc:~/mygit$ cat 1.txt
master add version2.
feature-A add version1
fix bug-001
hioier@pc:~/mygit$ git add 1.txt
hioier@pc:~/mygit$ git commit -m "fix bug-001"
[bug-001 3a43b77] fix bug-001
 1 file changed, 1 insertion(+)
```

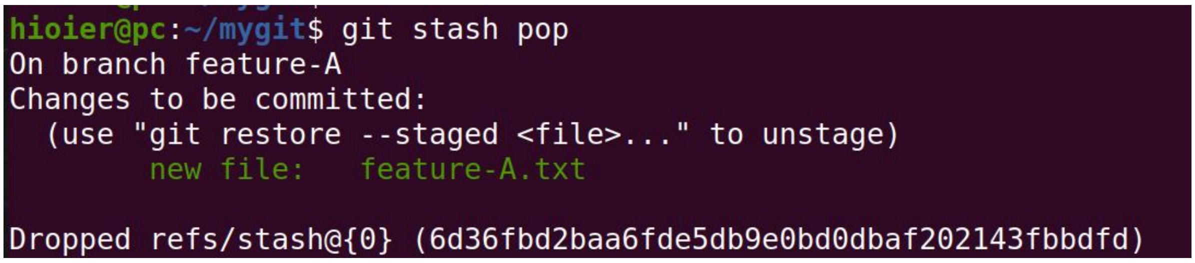


```
hioier@pc:~/mygit$ git merge --no-ff bug-001
Merge made by the 'recursive' strategy.
 1.txt | 1 +
 1 file changed, 1 insertion(+)
hioier@pc:~/mygit$ cat 1.txt
master add version2.
feature-A add version1
fix bug-001
```



```
hioier@pc:~/mygit$ git branch -d bug-001
Deleted branch bug-001 (was 3a43b77).
hioier@pc:~/mygit$ git log --graph --pretty=oneline
*   aa179dbdd03017223f5c68194c2b1a6e96cbeea1 (HEAD -> master) Merge branch 'bug-001'
|\
| * 3a43b773bc9d4bec7dd63fc84d10839040d03894 fix bug-001
|/
*   c6daaf54113cdf82b1f423b5a70925b9211d5bb3 (feature-A) handle conflict
```

接下来回到feature-A恢复现场继续干活：



```
hioier@pc:~/mygit$ git stash pop
On branch feature-A
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   feature-A.txt

Dropped refs/stash@{0} (6d36fbd2baa6fde5db9e0bd0dbaf202143fbbdfd)
```