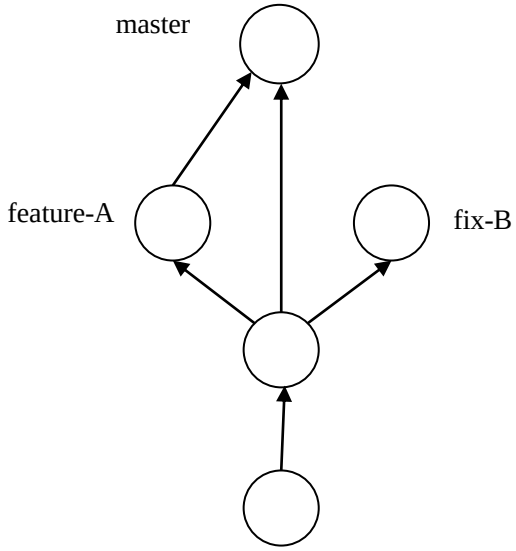
更改提交的操作

（1）git reset——回溯历史版本

通过前面学习的操作，我们已经学会如何在实现功能后进行提交，累积提交日志作为历史记录，借此不断培育一款软件。

Git 的另一特性便是可以灵活操作历史版本。借助分散仓库的优势，可以在不影响其他仓库的前提下对历史版本进行操作。

在这里，为了让各位熟悉对历史版本的操作，我们先回溯历史版本，创建一个名为 fix-B 的特性分支。

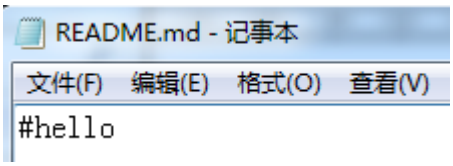


让我们先回溯到上一节 feature-A 分支创建之前，创建一个名为 fix-B 的特性分支。

要让仓库的 HEAD、暂存区、当前工作树回溯到指定状态，需要用到 git reset --hard 命令。只要提供目标时间点的哈希值，就可以完全恢复至该时间点的状态。

```
LZT@LZT-PC MINGW64 ~/hello (master)
$ git reset --hard 347b98bd9ba08a13c3abfd1da56bf75d7b6ff47b
HEAD is now at 347b98b This is a test
```

我们已经成功回溯到特性分支（feature-A）创建之前的状态。由于所有文件都回溯到了指定哈希值对应的时间点上，README.md 文件的内容也恢复到了当时的状态。

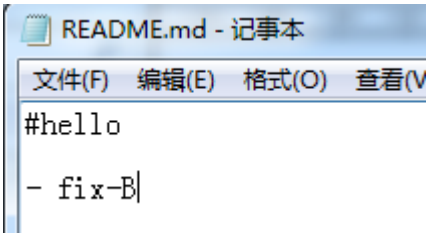


（2）创建 fix-B 分支

现在我们来创建 fix-B 特性分支。

```
LZT@LZT-PC MINGW64 ~/hello (master)
$ git checkout -b fix-B
Switched to a new branch 'fix-B'
```

作为这个主题的作业内容，我们在 README.md 文件中添加一行文字。

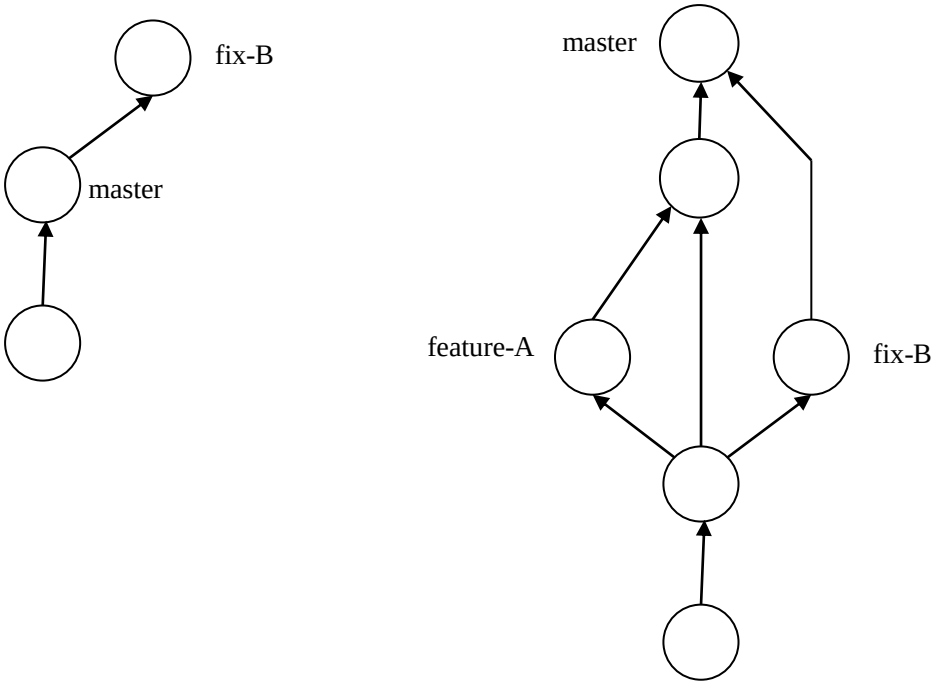


然后直接提交 README.md 文件。

```
LZT@LZT-PC MINGW64 ~/hello (fix-B)
$ git add README.md

LZT@LZT-PC MINGW64 ~/hello (fix-B)
$ git commit -m"Fix B"
[fix-B 9ed42be] Fix B
 1 file changed, 3 insertions(+), 1 deletion(-)
```

现在的状态如左图所示。接下来我们的目标是右图中所示的状态，即主干分支合并 feature-A 分支的修改后，又合并了 fix-B 的修改。



（3）推进至 feature-A 分支合并后的状态

首先恢复至 feature-A 分支合并后的状态。不妨称这一操作为“推进历史”。

git log 命令只能查看以当前状态为终点的历史日志。所以这里要使用 git reflog 命令，查看当前仓库的操作日志。在日志中找出回溯历史之前的哈希值，通过 git reset --hard 命令恢复到回溯历史前的状态。

首先执行 git reflog 命令，查看当前仓库执行过的操作的日志。

```
LZT@LZT-PC MINGW64 ~/hello (fix-B)
$ git reflog
9ed42be (HEAD -> fix-B) HEAD@{0}: commit: Fix B
347b98b (master) HEAD@{1}: checkout: moving from master to fix-B
347b98b (master) HEAD@{2}: reset: moving to 347b98bd9ba08a13c3abfd1da56bf75d7b6f
f47b
d7a3995 HEAD@{3}: merge feature-A: Merge made by the 'recursive' strategy.
347b98b (master) HEAD@{4}: checkout: moving from feature-A to master
66b5c18 (feature-A) HEAD@{5}: checkout: moving from master to feature-A
347b98b (master) HEAD@{6}: checkout: moving from feature-A to master
66b5c18 (feature-A) HEAD@{7}: commit: Add feature-A
347b98b (master) HEAD@{8}: checkout: moving from master to feature-A
347b98b (master) HEAD@{9}: commit (initial): This is a test
```

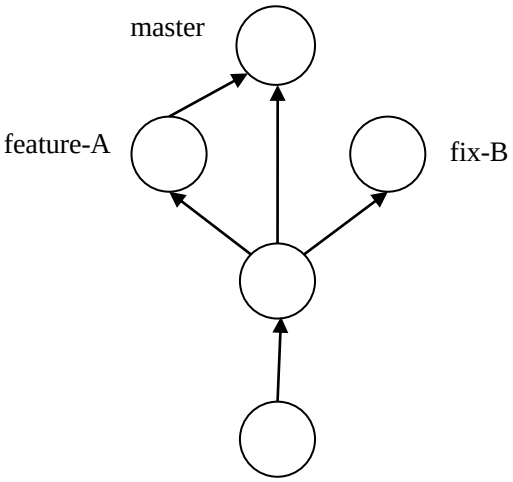
在日志中，我们可以看到 commit、checkout、reset、merge 等 Git 命令的执行记录。只要不进行 Git 的垃圾回收（GC，Garbage Collection），就可以通过日志随意调取近期的历史状态，就像给时间机器指定一个时间点，在过去未来中自由穿梭一般。即便开发者错误执行了 Git 操作，基本也都可以利用 git reflog 命令恢复到原先的状态。

从上面数第四行表示 featur-A 特性分支合并后的状态，对应哈希值为 d7a3995（哈希值只要输入 4 位以上就可以执行）。我们将 HEAD、暂存区、工作树恢复到这个时间点的状态。

```
LZT@LZT-PC MINGW64 ~/hello (fix-B)
$ git checkout master
Switched to branch 'master'

LZT@LZT-PC MINGW64 ~/hello (master)
$ git reset --hard d7a3995
HEAD is now at d7a3995 Merge branch 'feature-A'
```

之前我们使用 git reset --hard 命令回溯了历史，这里又再次通过它恢复到了回溯前的历史状态。当前的状态如图所示。



（4）消除冲突

现在只要合并 fix-B 分支，就可以得到我们想要的状态。让我们赶快进行合并操作。

```
LZT@LZT-PC MINGW64 ~/hello (master)
$ git merge --no-ff fix-B
Auto-merging README.md
CONFLICT (content): Merge conflict in README.md
Automatic merge failed; fix conflicts and then commit the result.
```

这时，系统告诉我们 README.md 文件发生了冲突（conflict）。系统在合并 README.md 文件时，feature-A 分支更改的部分与本次想要合并的 fix-B 分支更改的部分发生了冲突。不解决冲突就无法完成合并，所以我们打开 README.md 文件，解决这个冲突。

①查看冲突部分并将其解决

用编辑器打开 README.md 文件，就会发现其内容变成了下面这个样子。

```
#hello
<<<<<<< HEAD
- feature-A
=======

- fix-B
>>>>>>> fix-B
```

=======以上的部分是当前 HEAD 的内容，以下的部分是要合并的 fix-B 分支中的内容。我们在编辑器中将其改成想要的样子。

```
#hello

- feature-A
- fix-B
```

如上所示，本次修正让 featu-A 与 fix-B 的内容并存于文件之中。但是在实际的软件开发中，往往需要删除其中之一，所以各位在处理冲突时，务必要仔细分析冲突部分的内容后再进行修改。

②提交解决后的结果

冲突解决后，执行 git add 命令与 git commit 命令。

```
LZT@LZT-PC MINGW64 ~/hello (master|MERGING)
$ git add README.md

LZT@LZT-PC MINGW64 ~/hello (master|MERGING)
$ git commit -m"Fix conflict"
[master 73605bb] Fix conflict
```

由于本次更改解决了冲突，所以提交信息记为“Fix conflict”。

（5）git commiit --amend——修改提交信息

要修改上一条提交信息，可以使用 git commiit --amend 命令。

我们将上一条提交信息记为了“Fix conflict”，但它其实是 fix-B 分支的合并，解决合并时发生的冲突只是过程之一，这样标记实在不妥，于是，我们要修改这条提交信息。

```
LZT@LZT-PC MINGW64 ~/hello (master)
$ git commit --amend
```

执行上面的命令后，编辑器就会启动。

```
Fix conflict

# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
#
# Date:      Mon Nov 6 22:13:54 2017 +0800
#
# On branch master
# Changes to be committed:
#       modified:   README.md
#
```

编辑器中显示的内容如上所示，其中包含之前的提交信息。请将提交信息的部分修改为 Merge branch‘fix-B’，然后保存文件，关闭编辑器。（按 a，i 或 o 进入编辑模式，按 ESC 进入操作模式，在操作模式下，按两次大写的 Z 保存退出。）

```
Merge branch 'fix-B'

# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
#
# Date:      Mon Nov 6 22:13:54 2017 +0800
#
# On branch master
# Changes to be committed:
#       modified:   README.md
#
```

随后会显示下面这条结果。

```
LZT@LZT-PC MINGW64 ~/hello (master)
$ git commit --amend

[master f051357] Merge branch 'fix-B'
 Date: Mon Nov 6 22:13:54 2017 +0800
```

现在执行 git log --graph 命令，可以看到提交日志中的相应内容也已经被修改。

```
LZT@LZT-PC MINGW64 ~/hello (master)
$ git log --graph
*   commit 413fabeee57911125a487d1d3946610f94da39d6 (HEAD -> master)
|\  Merge: dc8ebf9 7cbf589
| | Author: ZitingLou <louziting@163.com>
| | Date:   Thu Nov 9 21:16:50 2017 +0800
| |
| |     Merge baranch 'fix-B'
| |
| * commit 7cbf589a79b87d4fb51f7f6f4fb997b832064433 (fix-B)
| | Author: ZitingLou <louziting@163.com>
| | Date:   Thu Nov 9 21:14:13 2017 +0800
| |
| |     Fix B
| |
*   |   commit dc8ebf9ab049ef02ec0a95ebb4a763188a09e224
|\  \  Merge: 5fc19ee 7aee812
| |/  Author: ZitingLou <louziting@163.com>
|/|   Date:   Thu Nov 9 21:11:32 2017 +0800
| |
| |       Merge branch 'feature-A'
| |
| * commit 7aee812c3ab9cdbd3100a64f36d9b56ba663f04e (feature-A)
|/  Author: ZitingLou <louziting@163.com>
|   Date:   Thu Nov 9 21:10:54 2017 +0800
|
|       Add feature-A
|
* commit 5fc19ee9ae1a3e03f8cf791f8b31aac77faf9e46
  Author: ZitingLou <louziting@163.com>
  Date:   Thu Nov 9 21:08:48 2017 +0800

      This is a test
```

（6）git rebase –i——压缩历史

在合并特性分支之前，我们发现已提交的内容中有些许拼写错误等，不妨提交一个修改，然后将这个修改包含到前一个提交之中，压缩成一个历史记录。这是个会经常用到的技巧，让我们来实际操作体会一下。

① 创建 feature-C 分支

首先，新建一个 feature-C 特性分支。

```
LZT@LZT-PC MINGW64 ~/hello (master)
$ git checkout -b feature-C
Switched to a new branch 'feature-C'
```

作为 feature-C 的功能实现，我们在 README.md 文件中添加一行文字，并且故意留下拼写错误，以便之后修正。

```
#hello

- feature-A
- fix-B
- faeture-C
```

提交这部分内容。这个小小的变更就没必要先执行 git add 命令，再执行 git commit 命

令了，我们用 git commit –am 命令来一次完成这两步操作。

```
LZT@LZT-PC MINGW64 ~/hello (feature-C)
$ git commit -am"Add feature-C"
[feature-C b236020] Add feature-C
 1 file changed, 1 insertion(+)
```

② 修正拼写错误

现在来修正刚才预留的拼写错误。请各位自行修正 README.md 文件的内容，修改后的差别如下所示：

```
LZT@LZT-PC MINGW64 ~/hello (feature-C)
$ git diff
diff --git a/README.md b/README.md
index fb9522b..f21eba0 100644
--- a/README.md
+++ b/README.md
@@ -3,4 +3,4 @@

 -feature-A
 -fix-B
--faeture-C
\ No newline at end of file
+-feature-C
\ No newline at end of file
```

然后进行提交

```
LZT@LZT-PC MINGW64 ~/hello (feature-C)
$ git commit -am "Fix typo"
[feature-C 7831906] Fix typo
 1 file changed, 1 insertion(+), 1 deletion(-)
```

错字漏字等失误称为 typo，所以我们将提交信息记为“Fix typo”。

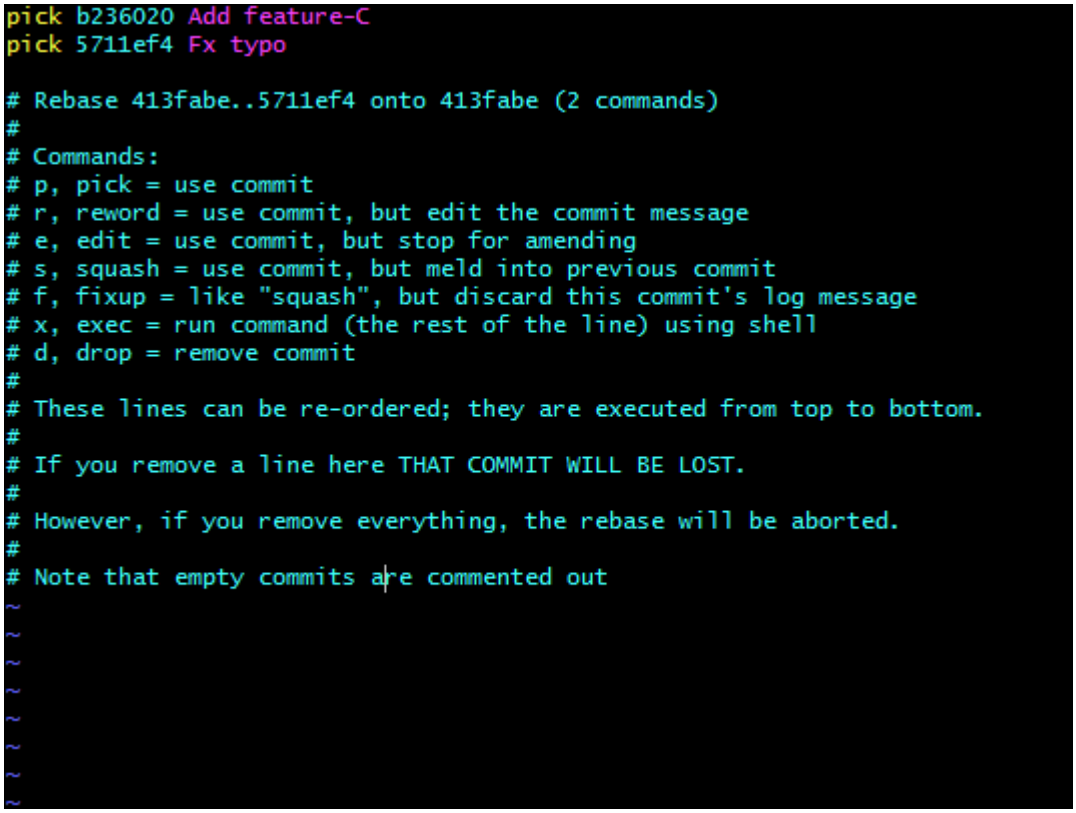
实际上，我们不希望在历史记录中看到这类提交，因为健全的历史记录并不需要它们。如果能在最初提交之前就发现并修正这些错误，也就不会出现这类提交了。

③ 更改历史

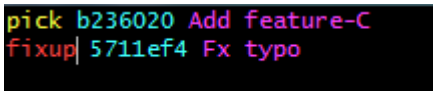
因此，我们来更改历史。将“Fix typo”修正的内容与之前一次的提交合并，在历史记录中合并为一次完美的提交。为此，我们要用到 git rebase 命令。

```
LZT@LZT-PC MINGW64 ~/hello (feature-C)
$ git rebase -i HEAD~2
```

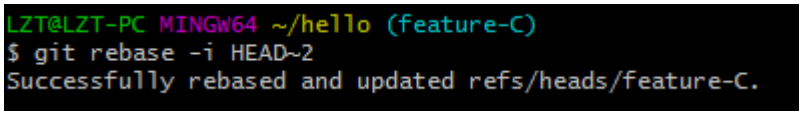
用上述方式执行命令，可以选定当前分支中包含 HEAD（最新提交）在内的两个最新历史记录为对象，并在编辑器中打开。



我们将 5711ef4 的 Fix typo 的历史记录压缩到 b236020 的 Add feature-C 里。按照下图所示，将 5711ef4 左侧的 pick 部分删除，改写为 fixup。



保存编辑器里的内容，关闭编辑器。



系统显示 rebase 成功。也就是两个提交改写成了一个新的提交。

现在再查看提交日志时会发现 Add feature-C 的哈希值已经不是 b236020 了，这证明提交已经被更改。

```
LZT@LZT-PC MINGW64 ~/hello (feature-C)
$ git log --graph
* commit 2a36f70a2373ea5c7a33720f6d86a83370b183c4 (HEAD -> feature-C)
| Author: ZitingLou <louziting@163.com>
| Date:   Thu Nov 9 21:21:00 2017 +0800
|
|     Add feature-C
|
*   commit 413fabeee57911125a487d1d3946610f94da39d6 (master)
|\  Merge: dc8ebf9 7cbf589
| | Author: ZitingLou <louziting@163.com>
| | Date:   Thu Nov 9 21:16:50 2017 +0800
| |
| |     Merge baranch 'fix-B'
| |
| * commit 7cbf589a79b87d4fb51f7f6f4fb997b832064433 (fix-B)
| | Author: ZitingLou <louziting@163.com>
| | Date:   Thu Nov 9 21:14:13 2017 +0800
| |
| |     Fix B
| |
```

这样一来，Fix typo 就从历史中被抹去，也就相当于 Add feature-C 中从来没有出现过拼写错误。这算是一种良性的历史改正。

④ 合并至 master 分支

feature-C 分支的使命告一段落，我们将它与 master 分支合并。

```
LZT@LZT-PC MINGW64 ~/hello (feature-C)
$ git checkout master
Switched to branch 'master'

LZT@LZT-PC MINGW64 ~/hello (master)
$ git merge --no-ff feature-C
Merge made by the 'recursive' strategy.
 README.md | 1 +
 1 file changed, 1 insertion(+)

LZT@LZT-PC MINGW64 ~/hello (master)
$ git log --graph
*   commit 3ed48d0b663551a5d2ed673aa4fa29d63bac1d3e (HEAD -> master)
|\  Merge: 413fabe 2a36f70
| | Author: ZitingLou <louziting@163.com>
| | Date:   Thu Nov 9 21:31:49 2017 +0800
| |
| |     Merge branch 'feature-C'
| |
| * commit 2a36f70a2373ea5c7a33720f6d86a83370b183c4 (feature-C)
|/  Author: ZitingLou <louziting@163.com>
|   Date:   Thu Nov 9 21:21:00 2017 +0800
|
|       Add feature-C
|
*   commit 413fabeee57911125a487d1d3946610f94da39d6
|\  Merge: dc8ebf9 7cbf589
| | Author: ZitingLou <louziting@163.com>
| | Date:   Thu Nov 9 21:16:50 2017 +0800
| |
| |     Merge baranch 'fix-B'
| |
| * commit 7cbf589a79b87d4fb51f7f6f4fb997b832064433 (fix-B)
| | Author: ZitingLou <louziting@163.com>
| | Date:   Thu Nov 9 21:14:13 2017 +0800
| |
| |     Fix B
```

master 分支整合了 feature-C 分支。开发进展顺利。