

Git 版本控制系列：使用 `git merge --no-ff` 合并分支

<https://blog.csdn.net/wangqingchuan92/article/details/103137960>

0x00 前言

文章中的文字可能存在语法错误以及标点错误，请谅解；

如果在文章中发现代码错误或其它问题请告知，感谢！

演示运行系统环境：Windows 10 家庭中文版， 64 位

Git 版本：git version 2.23.0.windows.1

0x01 `git merge` 和 `git merge --no-ff`

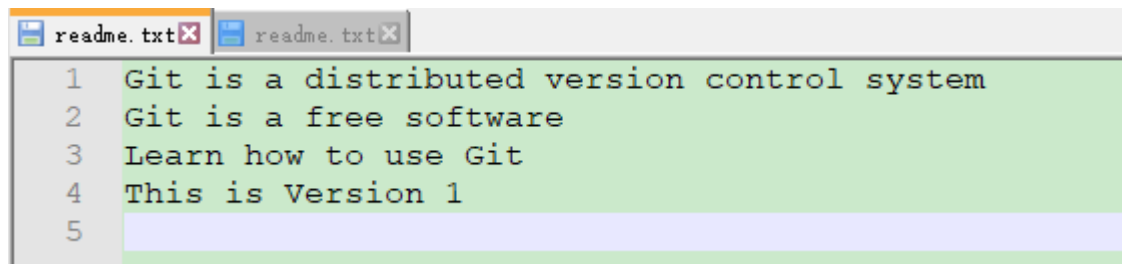
`git merge` 和 `git merge --no-ff` 都是合并分支，但是意义稍有不一致。`git merge` 也可以写成 `git merge --ff`，其中参数 `--ff` 意为 fast-forward。该命令指的是把 HEAD 指针指向要合并分支的头，完成一次合并。`git merge --no-ff` 中的 `--no-ff` 意为强行关掉 fast-forward，所以在使用这种方式后，分支合并后会生成一个新的 commit，这样，在使用 `git log` 从提交历史上就可以看到分支信息。

本文内容着重讲解 `git merge --no-ff` 合并分支，关于使用 `git merge` 合并分支的讲解可以参看我上一篇文档：

<https://blog.csdn.net/wangqingchuan92/article/details/103078680>

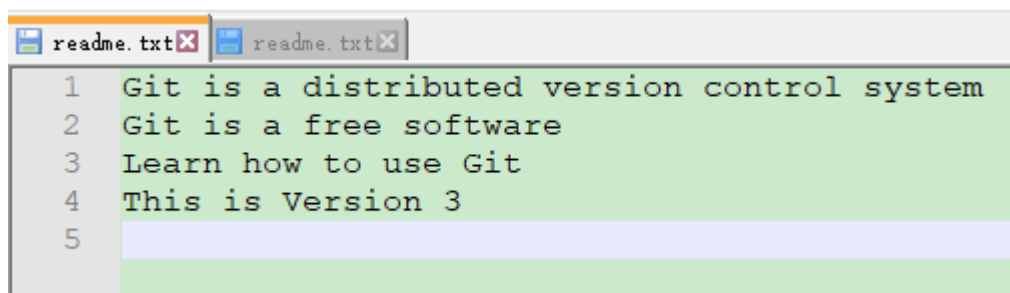
0x03 `git merge --no-ff` 合并分支举例

在 `git` 初始化 (`git init`) 后，新建一个 `readme.txt` 文件（注意 `readme.txt` 是在工作区），在文件中添加内容，并提交到暂存区（`git add readme.txt`）最后提交到版本库（`git commit -m "第一次提交"`）：



```
1 Git is a distributed version control system
2 Git is a free software
3 Learn how to use Git
4 This is Version 1
5
```

现在我们在 readme.txt 修改第 4 行“Version 2”并继续提交以及“Version 3”并继续提交后，所以当前工作区、暂存区、版本库中最新的 readme.txt 内容应该是：



```
1 Git is a distributed version control system
2 Git is a free software
3 Learn how to use Git
4 This is Version 3
5
```

git 版本提交历史如下：

```
时代凯撒@Clancey_Wang MINGW64 /e/GitHub/TestProject (master)
$ git log
commit 30500b71c71f7048f762fe5d4b8f1c3118db3879 (HEAD -> master)
Author: 麒麟川 <wangqingchuan92@126.com>
Date: Tue Nov 19 13:25:59 2019 +0800

    第三次提交

commit 01a37cffa1e8d83996627dfb1cbaed44bf6d1ff5
Author: 麒麟川 <wangqingchuan92@126.com>
Date: Tue Nov 19 13:25:28 2019 +0800

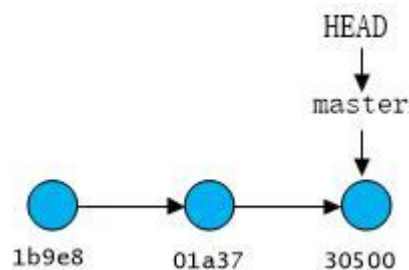
    第二次提交

commit 1b9e81705275ec5921e0247432aeeb8475064a43
Author: 麒麟川 <wangqingchuan92@126.com>
Date: Tue Nov 19 13:25:01 2019 +0800

    第一次提交
```

<https://blog.csdn.net/wangqingchuan92>

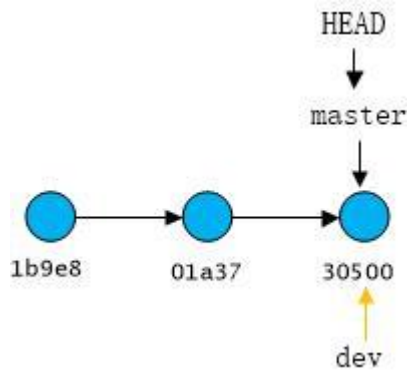
现在我们可以画一个 readme.txt 版本提交历史的示意图：



此时我们可以创建分支，分支名称为“dev”，然后将 HEAD 指针指向该分支，可以使用一条指令完成：git checkout -b dev：

```
时代凯撒@Clancey_Wang MINGW64 /e/GitHub/TestProject (dev)
$ git branch
* dev
  master
```

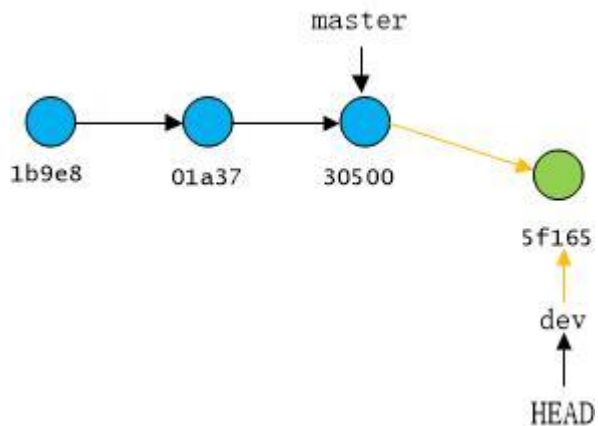
可以看到 HEAD 已经指向了 dev 分支，在示意图中的显示为：



现在就可以在 dev 分支上进行修改和提交了，我们对 readme.txt 进行修改，在第五行增加“dev OK”：

```
readme.txt x  readme.txt x
1  Git is a distributed version control system
2  Git is a free software
3  Learn how to use Git
4  This is Version 3
5  dev OK
```

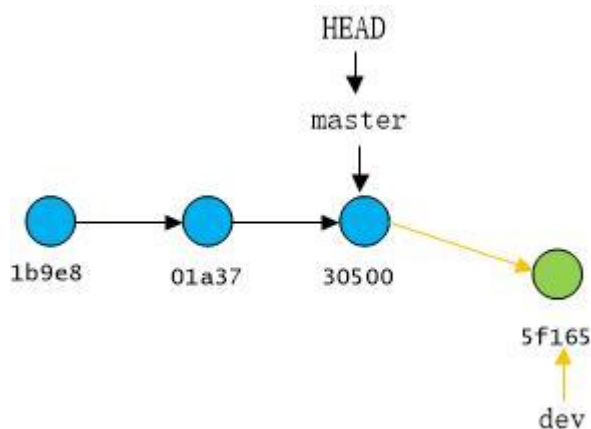
然后提交到暂存区（`git add readme.txt`）再提交到版本库（`git commit -m "dev 第一次提交"`）中，此时的 HEAD 指针位置如下所示：



在提交完成之后，我们将 HEAD 指针移动到 mater 主线分支（git checkout master），此时查看 readme.txt，可以发现这是第三次提交的版本：

```
readme.txt x  readme.txt x
1  Git is a distributed version control system
2  Git is a free software
3  Learn how to use Git
4  This is Version 3
5
```

这是因为那个添加“dev OK”的那个提交是在 dev 分支上，所以我们的 master 主线分支此时的提交点没有变化，所以 readme.txt 也没有改动：



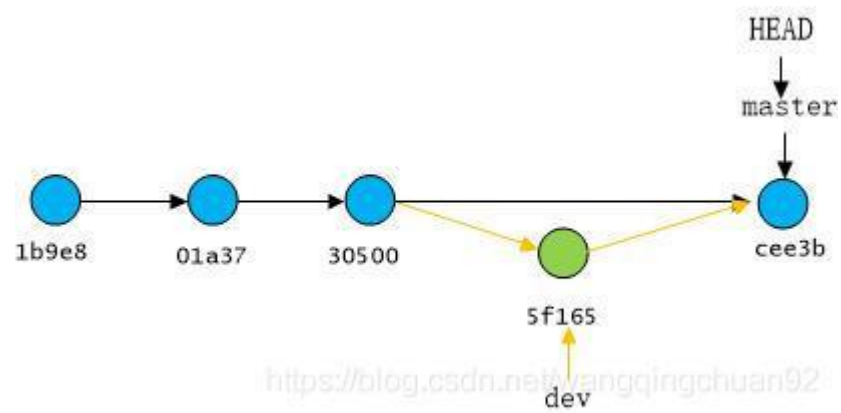
此时，我们想要将 dev 分支合并到 master 主线分支上（注意 HEAD 现在指向的是 master），而且想要保存之前的分支历史，则使用 `git merge --no-ff` 来合并，使用该命令合并时会创建一个新的 commit，所以加上 -m 参数，把 commit 描述写进去：`git merge --no-ff -m "merge with no ff" dev`:

```
时代凯撒@Clancey_Wang MINGW64 /e/GitHub/TestProject (master)
$ git merge --no-ff -m "merge with no ff" dev
Merge made by the 'recursive' strategy.
 readme.txt | 1 +
 1 file changed, 1 insertion(+)
```

现在用 `git log --graph --pretty=oneline --abbrev-commit` 查看提交历史：

```
时代凯撒@Clancey_Wang MINGW64 /e/GitHub/TestProject (master)
$ git log --graph --pretty=oneline --abbrev-commit
*   cee3b1c (HEAD -> master) merge with no ff
| \
|  * 5f165a1 (dev) dev第一次提交
| /
* 30500b7 第三次提交
* 01a37cf 第二次提交
* 1b9e817 第一次提交
```

在示意图中的显示为：



以上。

参考文档：

- 1.<https://segmentfault.com/q/1010000002477106>
 - 2.<https://www.liaoxuefeng.com/wiki/896043488029600/900005860592480>
 - 3.<https://blog.csdn.net/wangqingchuan92/article/details/103078680>
-