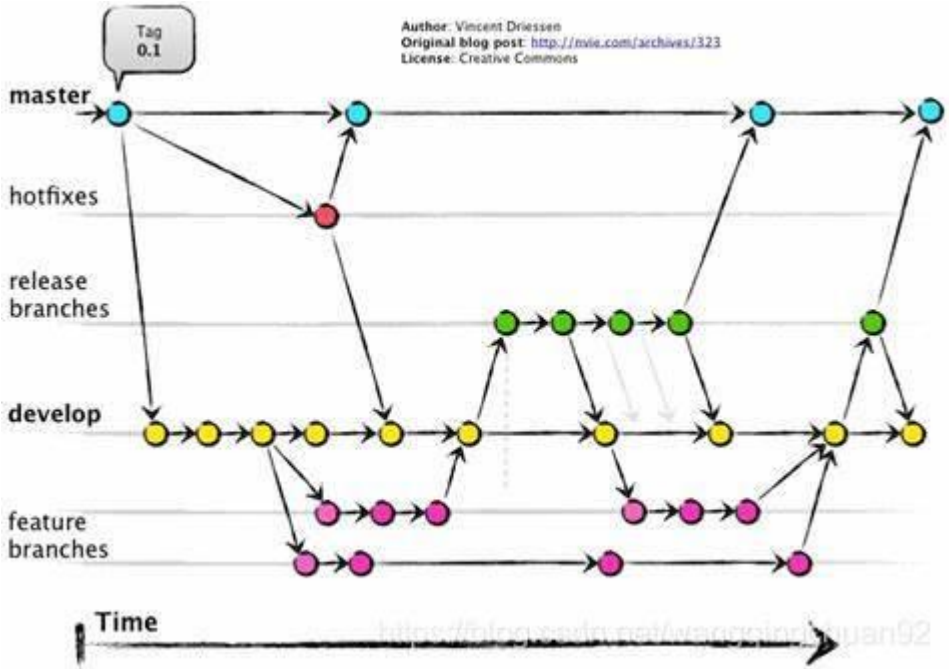
Git 版本控制系列：创建分支和分支合并
https://blog.csdn.net/wangqingchuan92/article/details/103078680
0x00 前言
文章中的文字可能存在语法错误以及标点错误，请谅解；

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 branch、git chekout 以及 git merge
在软件开发的过程中，可能会存在多人同时对软件进行编程开发或修改，这样则会导致最后出现多个版本的情况，造成版本管理混乱，所以这时候就需要分支管理了。分支(branches ）就是在软件的开发或修改记录的整体流程（主线）上进行分叉，可以根据需求有多个分叉，在对分叉后的分支进行操作不会影响到其它分支，当然在分支上开发修改完成之后可以再合回到主线，这样的好处除了可以便于提高开发效率之外，还有就是可以保证主线代码的完整性和可用性，保障主线代码是稳定代码。



使用 Git 进行版本控制的典型工作流程，注意看它的分支使用情况
（源自 https://arcadsoftware.com/news-events/blog/what-is-source-code-management/）

对于分支的创建和合并，我们一般会用到 git branch、 git checkout 和 git merge 命令。git branch 命令用于列出，创建或删除分支；git checkout 命令用于切换分支或恢复工作区文件；git merge 命令用于将两个或两个以上的版本合并一起 ；需要注意的是，这三个命令其后加的参数不同，所表达意义也不同。比如 git checkout -- hello.c 意思是将暂存区文件 hello.c 恢复到工作区内，git checkout master 意思是将当前 HEAD 指针指向 master 主线上。

0x02 创建分支以及分支合并举例
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"）：

readme.txt

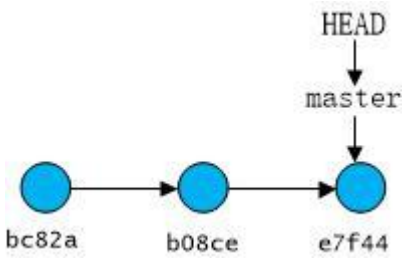
```
1  Git is a distributed version control system
2  Git is a free software
3  Learn how to use Git
4  This is Version 1
```

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、版本库中中最新的 readme.txt 内容应该是：

readme.txt

```
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
```

此时我们可以画一个 readme.txt 版本提交历史的示意图：



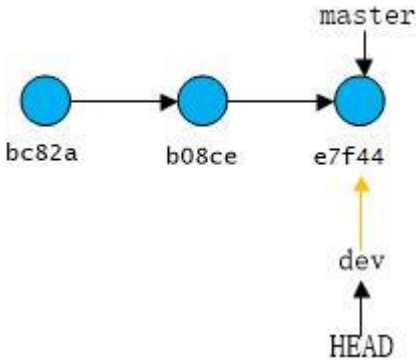
当前 HEAD 以及 master 指针指向“第三次提交”（即“This is version 3”的 readme.txt）版本。所以，每次的版本提交，master 主线都会向前移动一步，随着不断提交新版本，masterd 主线也会越来越长。

我们现在开始创建分支，分支的名称定为“dev”，则使用指令：git branch dev,然后让 HEAD 指针转到该 dev 分支：git checkout dev，然后使用 git branch 查看

当前 HEAD 指针定位位置（或者使用 git checkout -b dev 直接创建并指向 dev 分支）：

```
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：

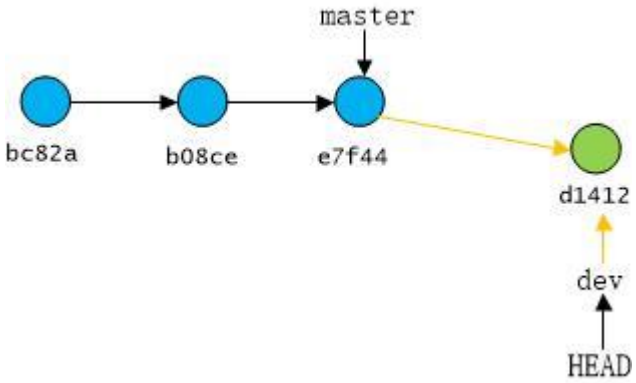


现在就可以在 dev 分支上进行修改和提交了，我们对 readme.txt 进行修改，在第五行增加“dev OK“：

readme.txt

```
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：

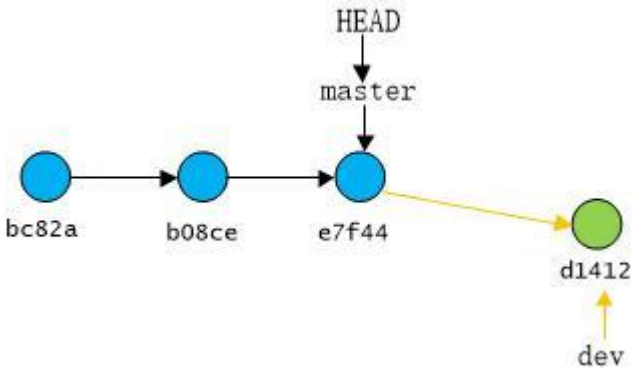


提交完成之后，我们切换到 master 主线分支上（git checkout master），然后我们再查看一下 readme.txt 文件，发现内容仍为第三次提交版本，没有改变：

readme.txt

```
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
```

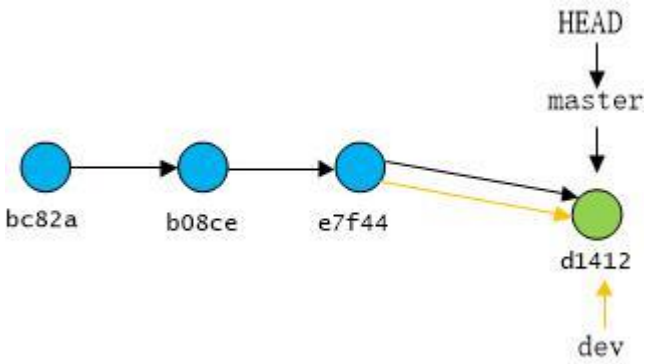
这是因为那个添加“dev OK”的那个提交是在 dev 分支上，所以我们的 master 主线分支此时的提交点没有变化，所以 readme.txt 也没有改动：



现在我们想要将 dev 分支的内容合并到主线分支上，使用 git merge <分支名>命令来进行合并 dev：git merge dev：

```
时代凯撒@Clancey_Wang MINGW64 /e/GitHub/TestProject (master)
$ git merge dev
Updating e7f4464..d141225
Fast-forward
 readme.txt | 3 ++-
 1 file changed, 2 insertions(+), 1 deletion(-)
```

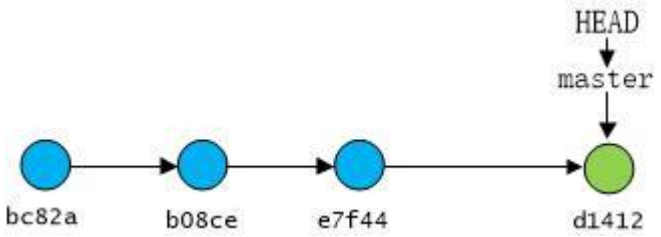
合并时出现的 Fast-forward 意思是合并的方式为“快进模式”，即将 master 指向 dev 分支上的最新提交：



需要注意的是，合并的方式有很多种，在这里选择了“快进模式”。

当我们合并之后，dev 分支对于我们就没有用处了，此时我们可以使用 git branch -d <分支名>来删除 dev 分支：git branch -d dev：

```
时代凯撒@Clancey_Wang MINGW64 /e/GitHub/TestProject (master)
$ git branch -d dev
Deleted branch dev (was d141225).
```



以上。

参考文档：

1.https://www.cnblogs.com/chenweichu/articles/5851876.html

2.https://www.liaoxuefeng.com/wiki/896043488029600/900003767775424

3.https://git-scm.com/book/zh/v2/Git-%E5%88%86%E6%94%AF-%E5%88%86%E6%94%AF%E7%9A%84%E6%96%B0%E5%BB%BA%E4%B8%8E%E5%90%88%E5%B9%B6
