**Git分支 - 基本分支操作（二）**

**6、新建一个分支并且使分支指向指定的提交对象**

使用命令：git branch branchname commitHash。

我们现在本地库中只有一个 master 分支，并且在 master 分支有三个提交历史。

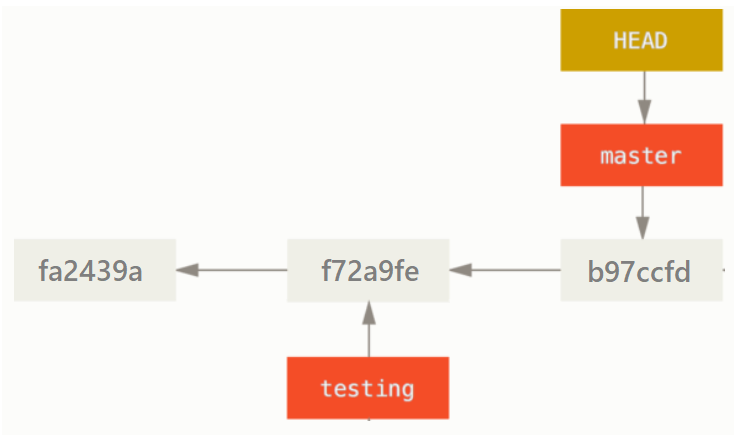
需求：创建一个 testing 分支，并且testing 分支指向 master 分支第二个版本。

|  |
| --- |
| Bash # 1.查看提交历史记录 L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/learngit (master) $ git log --graph --oneline \* b97ccfd (HEAD -> master) 第3次提交，新增内容：branch test v3 \* f72a9fe 第2次提交，新增内容：branch test v2 \* fa2439a 第1次提交，新增readme.txt文件  # 2.创建testing分支 L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/learngit (master) $ git branch testing f72a9fe  # 3.查看提交历史记录，此时我们还在master分支上 L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/learngit (master) $ git log --graph --oneline \* b97ccfd (HEAD -> master) 第3次提交，新增内容：branch test v3 \* f72a9fe (testing) 第2次提交，新增内容：branch test v2 \* fa2439a 第1次提交，新增readme.txt文件  # 4.切换到testing分支 L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/learngit (master) $ git checkout testing Switched to branch 'testing'  L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/learngit (testing) $ git log --graph --oneline \* f72a9fe (HEAD -> testing) 第2次提交，新增内容：branch test v2 \* fa2439a 第1次提交，新增readme.txt文件  # 5.再来查看readme.txt文件内容 L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/learngit (testing) $ cat readme.txt branch test v1 branch test v2 |

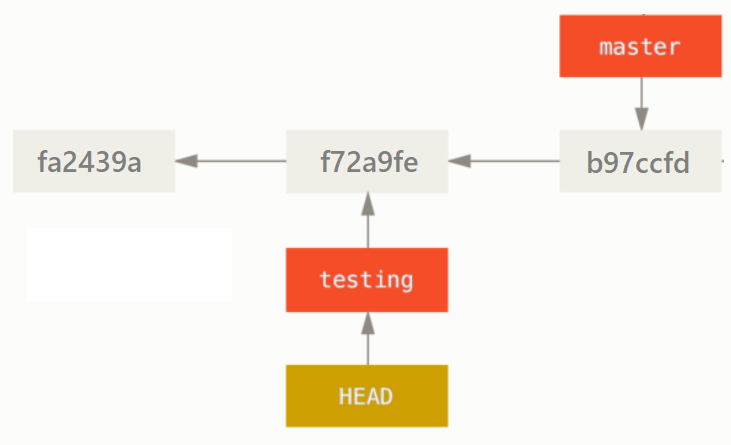
如上，我们看到新创建的 testing 分支，创建在master分支的第二个版本上了。

HEAD 指针的移动状态如下图：

执行git branch testing f72a9fe命令，创建testing 分支后：



执行git checkout testing命令，切换分之后：



**总结：**

在日常工作中，我们可以这样的操作，切换到之前的版本，或者其他人之前写过的版本。我们把自习需要的代码进行保存，然后就可以在切换到自己开发的分支上，应用这些代码，同时把刚才创建的分支删除掉即可。

（该操作在工作中非常的实用）

**7、思考：**

git checkout -b branchname commit-Hash命令，

能否代替下面两个命令：

* git branch branchname commitHash
* git checkout branchname

**答案是可以的。**

示例如下：

|  |
| --- |
| Bash # 1.查看本地库中所有的分支 L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/learngit (master) $ git branch \* master  # 2.查看当前分支的提交历史 L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/learngit (master) $ git log --oneline b97ccfd (HEAD -> master) 第3次提交，新增内容：branch test v3 f72a9fe 第2次提交，新增内容：branch test v2 fa2439a 第1次提交，新增readme.txt文件  # 3.创建testing分支，并切换 L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/learngit (master) $ git checkout -b testing f72a9fe Switched to a new branch 'testing'  # 4.查看testing分支的提交历史 L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/learngit (testing) $ git log --oneline f72a9fe (HEAD -> testing) 第2次提交，新增内容：branch test v2 fa2439a 第1次提交，新增readme.txt文件  # 5.查看readme.txt是否是V2版本 L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/learngit (testing) $ cat readme.txt branch test v1 branch test v2 |

**8、项目分叉历史的形成**

查看当前版本库的如上，此刻正在 testing 分支的上工作，在 readme.txt 文件在第二个版本。

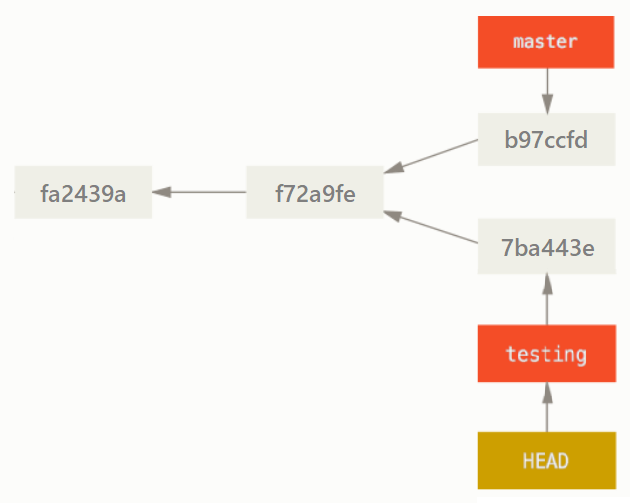
需求：我需要在 testing 分支的上，修改 readme.txt 文件，并提交新版本。

|  |
| --- |
| Bash # 1.查看版本库状态 # 分支情况 L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/learngit (testing) $ git branch  master \* testing  # 分支的提交历史 L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/learngit (testing) $ git log --oneline f72a9fe (HEAD -> testing) 第2次提交，新增内容：branch test v2 fa2439a 第1次提交，新增readme.txt文件  # 2.修改readme.txt文件并提交 L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/learngit (testing) $ echo "branch test v333 testing" >> readme.txt  L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/learngit (testing) $ git commit -a -m 'testing 分支的提交' warning: LF will be replaced by CRLF in readme.txt. The file will have its original line endings in your working directory [testing 7ba443e] testing 分支的提交  1 file changed, 1 insertion(+)  # 3.查看testing分支历史提交 L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/learngit (testing) $ git log --oneline --graph \* 7ba443e (HEAD -> testing) testing 分支的提交 \* f72a9fe 第2次提交，新增内容：branch test v2 \* fa2439a 第1次提交，新增readme.txt文件  # 4.查看所有分支的历史提交 L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/learngit (testing) $ git log --oneline --graph --all \* 7ba443e (HEAD -> testing) testing 分支的提交 | \* b97ccfd (master) 第3次提交，新增内容：branch test v3 |/ \* f72a9fe 第2次提交，新增内容：branch test v2 \* fa2439a 第1次提交，新增readme.txt文件 |

我们可以看到，现在这个项目的提交历史已经产生了分叉。

|  |
| --- |
| 在实际工作中，是因为你创建了一个新分支，并切换过去进行了一些工作，可能随后又切换回 master 分支进行了另外一些工作，这样针对的是不同分支进行改动，就会在工作目录中产生项目的分叉历史。  你可以在不同分支间不断地来回切换和工作，并在时机成熟时将它们合并起来。 而所有这些工作，你需要的命令只有 branch、checkout 和 commit。 |

此时 HEAD 指针的状态如下图：



**9、分支的总结**

由于 Git 的分支实质上仅是包含，所指对象校验和（长度为 40 的 SHA-1 值字符串）的文件，所以它的创建和销毁都异常高效。 创建一个新分支就相当于往一个文件中写入 41 个字节（40 个字符和 1 个换行符），如此的简单能不快吗。

这里简单说明一下：

|  |
| --- |
| Bash # 1.查看.git/refs/heads目录 L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/learngit (testing) $ ll .git/refs/heads/ total 2 -rw-r--r-- 1 L 197121 41 4月 16 22:07 master -rw-r--r-- 1 L 197121 41 4月 17 11:26 testing  # 可以看到.git/refs/heads目录存放的是版本库中的分支。  # 2.查看testing文件只呢个中的内容给 L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/learngit (testing) $ cat .git/refs/heads/testing 7ba443e4f6591631af4276bd3aa7356ff0f61fac  # 可以看到是一个提交的索引，对比上图中的`7ba443e`提交，你就明白了。  # 3.在往深说一点，查看.git/HEAD文件 L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/learngit (testing) $ cat .git/HEAD ref: refs/heads/testing  # .git/HEAD文件就是之前说的HEAD指针，指向了当前正在工作的分支。 # 是不是一切都非常的简单高效。 |

这与过去大多数版本控制系统形成了鲜明的对比，它们在创建分支时，将所有的项目文件都复制一遍，并保存到一个特定的目录。 完成这样繁琐的过程通常需要好几秒钟，有时甚至需要好几分钟。所需时间的长短，完全取决于项目的规模。

而在 Git 中，任何规模的项目都能在瞬间创建新分支。 同时，由于每次提交都会记录父对象，所以寻找恰当的合并基础（即共同祖先）也是同样的简单和高效。 这些高效的特性使得 Git 鼓励开发人员频繁地创建和使用分支。

作者：繁华似锦Fighting 链接：https://www.jianshu.com/p/e6b93f1f8c04 来源：简书 著作权归作者所有。商业转载请联系作者获得授权，非商业转载请注明出处。