

Git 实验 5 ： 创建和合并分支

制作人：陈晓华，齐美萍 qq: 78976932 微信号: chen-jeo

开源项目: <https://github.com/chenxhjeo>, 个人博客: <https://chenxhjeo.github.io>

初建日期: 2017.03.10 修订日期: 2017.03.12

一、 实验目的

- 1、实现分支的创建、合并
- 2、学习解决分支冲突。
- 3、了解分支管理策略。
- 4、掌握与远程仓库的分支交互。

二、 实验过程

在版本回退中,大家应该已经了解了每次提交,Git 都把它们串成一条时间线,这条时间线就是一个分支。

在之前的实验中,我们的操作只有一条时间线,在 Git 里,这个分支叫主分支,即 master 分支。HEAD 严格来说不是指向提交,而是指向 master, master 才是指向提交的,所以,HEAD 指向的就是当前分支。下面我们一起来实践一下如何创建和合并分支。

1、实现分支的创建、合并

- (1) 首先,进入先前创建的本地仓库 testgit 目录中(若先前没有创建本地仓库,请从实验一开始再来一次),然后 `git checkout -b 分支名` 创建分支(分支名可随意定),切换到 dev 分支上。如下操作:



```
MINGW64:/d/workspaces/Git/www/testgit

Qmp@Qimp MINGW64 ~
$ cd D:/workspaces/Git/www/testgit

Qmp@Qimp MINGW64 /d/workspaces/Git/www/testgit (master)
$ git checkout -b dev
Switched to a new branch 'dev'

Qmp@Qimp MINGW64 /d/workspaces/Git/www/testgit (dev)
$
```

The screenshot shows a Windows terminal window with the title 'MINGW64:/d/workspaces/Git/www/testgit'. The user is in the 'testgit' directory. They run the command `git checkout -b dev`, which creates a new branch named 'dev' and switches to it. The terminal output shows the prompt changing from '(master)' to '(dev)'. A red box highlights the command `git checkout -b dev`, and a red arrow points to the text '创建并切换分支' (Create and switch branch).

** `git checkout` 命令加上 `-b` 参数表示创建并切换,相当于 2 条命令:

```
git branch dev
git checkout dev
```

- (2) 用 `git branch` 命令查看自己当前目录下所有的分支,此时可查看到自己创建的分支,当前分支前面会添加一个“*”:

```
Qmp@Qimp MINGW64 /d/workspaces/Git/www/testgit (dev)
$ git branch
* dev
  master

Qmp@Qimp MINGW64 /d/workspaces/Git/www/testgit (dev)
$ |
```

- (3) 我们在 dev 分支上继续做 demo，比如我们现在在 readme.txt 再增加一行 7777777。首先我们先用 cat 来查看下 readme.txt 内容，接着添加内容 7777777，如下：

```
Qmp@Qimp MINGW64 /d/workspaces/Git/www/testgit (dev)
$ cat readme.txt
1111111
2222222
3333333
4444444
6666666

Qmp@Qimp MINGW64 /d/workspaces/Git/www/testgit (dev)
$ vi readme.txt

Qmp@Qimp MINGW64 /d/workspaces/Git/www/testgit (dev)
$ cat readme.txt
1111111
2222222
3333333
4444444
6666666
7777777

Qmp@Qimp MINGW64 /d/workspaces/Git/www/testgit (dev)
$ git add readme.txt

Qmp@Qimp MINGW64 /d/workspaces/Git/www/testgit (dev)
$ git commit -m "dev分支上增加内容7777777"
[dev 2578120] dev分支上增加内容7777777
1 file changed, 2 insertions(+), 1 deletion(-)

Qmp@Qimp MINGW64 /d/workspaces/Git/www/testgit (dev)
$ |
```

【补充】vi 编辑器使用方法：vi 文件名 → i → 内容编辑 → Esc 按键 → 输入 :wq → 回车键保存退出。

- (4) 现在 dev 分支工作已完成，现在我们用 git checkout 切换到主分支 master 上，继续查看 readme.txt 内容如下：

```
Qmp@Qimp MINGW64 /d/workspaces/Git/www/testgit (dev)
$ git checkout master
Switched to branch 'master'

Qmp@Qimp MINGW64 /d/workspaces/Git/www/testgit (master)
$ cat readme.txt
1111111
2222222
3333333
4444444
6666666

Qmp@Qimp MINGW64 /d/workspaces/Git/www/testgit (master)
$ |
```

- (5) 现在我们可以把 dev 分支上的内容合并到分支 master 上了，可以在 master 分支上，使用如下命令 git merge dev 如下所示：

```
Qmp@Qimp MINGW64 /d/workspaces/Git/www/testgit (master)
$ git merge dev
Updating f256f29..2578120
Fast-forward
 readme.txt | 3 ++-
 1 file changed, 2 insertions(+), 1 deletion(-)

Qmp@Qimp MINGW64 /d/workspaces/Git/www/testgit (master)
$ cat readme.txt
1111111
2222222
3333333
4444444
6666666
7777777

Qmp@Qimp MINGW64 /d/workspaces/Git/www/testgit (master)
$
```

`git merge` 命令用于合并指定分支到当前分支上,合并后,再查看 `readme.txt` 内容,可以看到,和 `dev` 分支最新提交的是完全一样的。

- (6) 注意到上面的 Fast-forward 信息,Git 告诉我们,这次合并是“快进模式”,也就是直接把 `master` 指向 `dev` 的当前提交,所以合并速度非常快。合并完成后,我们可以接着删除 `dev` 分支了,操作如下:

```
Qmp@Qimp MINGW64 /d/workspaces/Git/www/testgit (master)
$ git branch -d dev
Deleted branch dev (was 2578120).

Qmp@Qimp MINGW64 /d/workspaces/Git/www/testgit (master)
$ git branch
* master

Qmp@Qimp MINGW64 /d/workspaces/Git/www/testgit (master)
$
```

2、解决分支冲突

当两条分支对同一个文件的同一个文本块进行了不同的修改,并试图合并时,Git 不能自动合并的,称之为**冲突**(conflict)。解决冲突需要人工处理。

- (1) 下面我们还是一步一步来,先新建一个新分支,比如名字叫 `fenzhi`,在 `readme.txt` 添加一行内容 `8888888`,然后提交,如下所示:

```
Qmp@Qimp MINGW64 /d/workspaces/Git/www/testgit (master)
$ git checkout -b fenzhi
Switched to a new branch 'fenzhi'

Qmp@Qimp MINGW64 /d/workspaces/Git/www/testgit (fenzhi)
$ cat readme.txt
1111111
2222222
3333333
4444444
6666666
7777777

Qmp@Qimp MINGW64 /d/workspaces/Git/www/testgit (fenzhi)
$ vi readme.txt

Qmp@Qimp MINGW64 /d/workspaces/Git/www/testgit (fenzhi)
$ cat readme.txt
1111111
2222222
3333333
4444444
6666666
7777777
8888888

Qmp@Qimp MINGW64 /d/workspaces/Git/www/testgit (fenzhi)
$ git add readme.txt

Qmp@Qimp MINGW64 /d/workspaces/Git/www/testgit (fenzhi)
$ git commit -m "添加内容8888888"
[fenzhi ble718c] 添加内容8888888
1 file changed, 1 insertion(+)

Qmp@Qimp MINGW64 /d/workspaces/Git/www/testgit (fenzhi)
$
```

- (2) 接着，我们现在切换到 master 分支上来，也在最后一行添加内容，内容为 9999999，如下所示（和 fenzhi 分支的操作相同）：

```
Qmp@Qimp MINGW64 /d/workspaces/Git/www/testgit (fenzhi)
$ git checkout master
Switched to branch 'master'

Qmp@Qimp MINGW64 /d/workspaces/Git/www/testgit (master)
$ cat readme.txt
1111111
2222222
3333333
4444444
6666666
7777777

Qmp@Qimp MINGW64 /d/workspaces/Git/www/testgit (master)
$ vi readme.txt

Qmp@Qimp MINGW64 /d/workspaces/Git/www/testgit (master)
$ cat readme.txt
1111111
2222222
3333333
4444444
6666666
7777777
9999999

Qmp@Qimp MINGW64 /d/workspaces/Git/www/testgit (master)
$ git add readme.txt

Qmp@Qimp MINGW64 /d/workspaces/Git/www/testgit (master)
$ git commit -m "在master分支上新增内容9999999"
[master f04637f] 在master分支上新增内容9999999
1 file changed, 1 insertion(+)

Qmp@Qimp MINGW64 /d/workspaces/Git/www/testgit (master)
$ |
```

- (3) 现在我们需要在 master 分支上来合并 fenzhi，如下操作：

```
Qmp@Qimp MINGW64 /d/workspaces/Git/www/testgit (master)
$ git merge fenzhi → 在master分支上合并fenzhi
Auto-merging readme.txt
CONFLICT (content): Merge conflict in readme.txt 产生冲突
Automatic merge failed; fix conflicts and then commit the result.

Qmp@Qimp MINGW64 /d/workspaces/Git/www/testgit (master|MERGING)
$ git status → 查看状态
On branch master
You have unmerged paths.
  (fix conflicts and run "git commit")
  (use "git merge --abort" to abort the merge)

Unmerged paths:
  (use "git add <file>..." to mark resolution)

        both modified:   readme.txt

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        qi.txt
        softwareEngineer/
        testgit2/

no changes added to commit (use "git add" and/or "git commit -a")

Qmp@Qimp MINGW64 /d/workspaces/Git/www/testgit (master|MERGING)
$ cat readme.txt → 查看readme.txt
1111111
2222222
3333333
4444444
6666666
7777777
<<<<<< HEAD
9999999
=====
8888888
>>>>>> fenzhi → 冲突代码

Qmp@Qimp MINGW64 /d/workspaces/Git/www/testgit (master|MERGING)
$ |
```

Git 用<<<<<<, =====, >>>>>>标记出不同分支的内容，其中<<<HEAD 是指主分支修改的内容，>>>>>fenzhi 是指 fenzhi 上修改的内容。提示中告诉我们 master 分支和 fenzhi 分支在同一行进行了修改，master 分支上为

9999999, 而 fenzhi 分支上是 8888888, 将 fenzhi 分支上的内容修改为与主分支相同的内容。

(4) 修改冲突部分内容, 重新提交 (add → commit), 最后可用 `git log` 查看分支合并情况。

3、分支管理策略

分支策略: 首先 master 主分支应该是非常稳定的, 也就是用来发布新版本, 一般情况下不允许在上面干活, 而是选择在新建的 dev 分支上干活, 干完活后合并到主分支 master 上, 或者说 dev 分支代码稳定后再合并到主分支 master 上。

合并分支时, git 一般使用 "Fast forward" 模式, 在这种模式下, 删除分支后, 会丢掉分支信息, 现在我们来使用带参数 `--no-ff` 来禁用 "Fast-forward" 模式。步骤如下:

1. 创建一个 dev 分支。
2. 修改 readme.txt 内容。
3. 添加到暂存区。
4. 切换回主分支(master)。
5. 合并 dev 分支, 使用命令 `git merge --no-ff -m "注释" dev`。
6. 查看历史记录。

4、将新分支推送到 github 以及删除 github 上的分支

- 推送主分支:

```
git init
git add a.txt
git commit -m "first commit"
git remote add origin https://github.com/chenxhjeo/learngit.git
git push -u origin master
```

上面命令将本地的 master 分支推送到 origin 主机, 同时指定 origin 为默认主机, 后面就可以不加任何参数使用 `git push` 了。

- 推送和删除分支:

在本地新建一个分支: `git branch dev`

切换到你的新分支: `git checkout dev`

将新分支发布在 github 上: `git push origin dev`

在本地删除一个分支: `git branch -d dev`

在 github 远程端删除一个分支: `git push origin :dev` (分支名前的冒号代表删除)

5、从远程的分支获取最新的版本

Git 中从远程的分支获取最新的版本到本地有这样 2 个命令:

1) `git fetch`: 相当于是从远程获取最新版本到本地, 不会自动 merge

```
git fetch origin master
git log -p master..origin/master
git merge origin/master
```

以上命令的含义: 首先从远程的 origin 的 master 主分支下载最新的版本到 origin/master 分支上, 然后比较本地的 master 分支和 origin/master 分支的差别, 最后进行合并。上述过程可以用以下更清晰的方式进行:

```
git fetch origin master:tmp
git diff tmp
git merge tmp
```

从远程获取最新的版本到本地的 test 分支上，之后再进行比较合并

2) git pull: 相当于是从远程获取最新版本并 merge 到本地

```
git pull origin master
```

上述命令其实相当于 git fetch 和 git merge

在实际使用中, git fetch 更安全一些, 因为在 merge 前, 我们可以查看更新情况, 然后再决定是否合并

三、 实验总结

1、总结创建与合并分支命令如下:

查看分支: `git branch`

创建分支: `git branch 分支名`

切换分支: `git checkout 分支名`

创建+切换分支: `git checkout -b 分支名`

合并某分支到当前分支: `git merge 分支名`

删除分支: `git branch -d 分支名`

2、除上述的直接**合并方式**外外, 还有 2 中合并方式:

压力合并: `git checkout master`

```
git merge --squash dev
```

此时, dev 上的所有提交已经合并到当前工作区并暂存, 但还没有作为一个提交, 可以像其他提交一样, 把这个改动提交到版本库中:

```
git commit -m "something from dev"
```

拣选合并: 比如在 dev 上的某个提交叫: 321d76f, 把它合并到 master 中:

```
git checkout master
```

```
git cherry-pick 321d76f
```

要拣选多个提交, 可以给 git cherry-pick 命令传递 -n 选项, 比如:

```
git cherry-pick -n 321d76f
```

注: 以上内容在 PDF 格式显示时, 换页会出现大片空缺。