# Git 分支管理

几乎每一种版本控制系统都以某种形式支持分支。使用分支意味着你可以从开发主线上分离开来，然后在不影响主线的同时继续工作。

有人把 Git 的分支模型称为**必杀技特性**，而正是因为它，将 **Git** 从版本控制系统家族里区分出来。

创建分支命令：

git branch (branchname)

切换分支命令:

git checkout (branchname)

当你切换分支的时候，Git 会用该分支的最后提交的快照替换你的工作目录的内容， 所以多个分支不需要多个目录。

合并分支命令:

git merge

你可以多次合并到统一分支， 也可以选择在合并之后直接删除被并入的分支。

开始前我们先创建一个测试目录：

$ mkdir gitdemo

$ cd gitdemo/

$ git initInitialized empty Git repository...

$ touch README

$ git add README

$ git commit -m '第一次版本提交'[master (root-commit) 3b58100] 第一次版本提交

1 file changed, 0 insertions(+), 0 deletions(-)

create mode 100644 README

## Git 分支管理

### **列出分支**

列出分支基本命令：

git branch

没有参数时，**git branch** 会列出你在本地的分支。

$ git branch\* master

此例的意思就是，我们有一个叫做 **master** 的分支，并且该分支是当前分支。

当你执行 **git init** 的时候，缺省情况下 Git 就会为你创建 **master** 分支。

如果我们要手动创建一个分支。执行 **git branch (branchname)** 即可。

$ git branch testing

$ git branch\* master

testing

现在我们可以看到，有了一个新分支 **testing**。

当你以此方式在上次提交更新之后创建了新分支，如果后来又有更新提交， 然后又切换到了 **testing** 分支，Git 将还原你的工作目录到你创建分支时候的样子。

接下来我们将演示如何切换分支，我们用 git checkout (branch) 切换到我们要修改的分支。

$ ls

README

$ echo 'runoob.com' > test.txt

$ git add .

$ git commit -m 'add test.txt'[master 3e92c19] add test.txt

1 file changed, 1 insertion(+)

create mode 100644 test.txt

$ ls

README        test.txt

$ git checkout testingSwitched to branch 'testing'

$ ls

README

当我们切换到 **testing** 分支的时候，我们添加的新文件 test.txt 被移除了。切换回 **master** 分支的时候，它们有重新出现了。

$ git checkout masterSwitched to branch 'master'

$ ls

README        test.txt

我们也可以使用 git checkout -b (branchname) 命令来创建新分支并立即切换到该分支下，从而在该分支中操作。

$ git checkout -b newtestSwitched to a new branch 'newtest'

$ git rm test.txt

rm 'test.txt'

$ ls

README

$ touch runoob.php

$ git add .

$ git commit -am 'removed test.txt、add runoob.php'[newtest c1501a2] removed test.txt、add runoob.php

2 files changed, 1 deletion(-)

create mode 100644 runoob.php

delete mode 100644 test.txt

$ ls

README        runoob.php

$ git checkout masterSwitched to branch 'master'

$ ls

README        test.txt

如你所见，我们创建了一个分支，在该分支的上移除了一些文件 test.txt，并添加了 runoob.php 文件，然后切换回我们的主分支，删除的 test.txt 文件又回来了，且新增加的 runoob.php 不存在主分支中。

使用分支将工作切分开来，从而让我们能够在不同开发环境中做事，并来回切换。

### **删除分支**

删除分支命令：

git branch -d (branchname)

例如我们要删除 testing 分支：

$ git branch\* master

testing

$ git branch -d testingDeleted branch testing (was 85fc7e7).

$ git branch\* master

### **分支合并**

一旦某分支有了独立内容，你终究会希望将它合并回到你的主分支。 你可以使用以下命令将任何分支合并到当前分支中去：

git merge

$ git branch\* master

newtest

$ ls

README        test.txt

$ git merge newtestUpdating 3e92c19..c1501a2Fast-forward

runoob.php | 0

test.txt | 1 -

2 files changed, 1 deletion(-)

create mode 100644 runoob.php

delete mode 100644 test.txt

$ ls

README        runoob.php

以上实例中我们将 newtest 分支合并到主分支去，test.txt 文件被删除。

合并完后就可以删除分支:

$ git branch -d newtestDeleted branch newtest (was c1501a2).

删除后， 就只剩下 master 分支了：

$ git branch\* master

### **合并冲突**

合并并不仅仅是简单的文件添加、移除的操作，Git 也会合并修改。

$ git branch\* master

$ cat runoob.php

首先，我们创建一个叫做 change\_site 的分支，切换过去，我们将 runoob.php 内容改为:

<?php

echo 'runoob';?>

创建 change\_site 分支：

$ git checkout -b change\_siteSwitched to a new branch 'change\_site'

$ vim runoob.php

$ head -3 runoob.php<?php

echo 'runoob';?>

$ git commit -am 'changed the runoob.php'[change\_site 7774248] changed the runoob.php

1 file changed, 3 insertions(+)

将修改的内容提交到 change\_site 分支中。 现在，假如切换回 master 分支我们可以看内容恢复到我们修改前的(空文件，没有代码)，我们再次修改 runoob.php 文件。

$ git checkout masterSwitched to branch 'master'

$ cat runoob.php

$ vim runoob.php # 修改内容如下

$ cat runoob.php<?php

echo 1;?>

$ git diff

diff --git a/runoob.php b/runoob.php

index e69de29..ac60739 100644--- a/runoob.php+++ b/runoob.php@@ -0,0 +1,3 @@+<?php+echo 1;+?>

$ git commit -am '修改代码'[master c68142b] 修改代码

1 file changed, 3 insertions(+)

现在这些改变已经记录到我的 "master" 分支了。接下来我们将 "change\_site" 分支合并过来。

$ git merge change\_siteAuto-merging runoob.php

CONFLICT (content): Merge conflict in runoob.phpAutomatic merge failed; fix conflicts and then commit the result.

$ cat runoob.php # 代开文件，看到冲突内容<?php<<<<<<< HEAD

echo 1;=======

echo 'runoob';>>>>>>> change\_site?>

我们将前一个分支合并到 master 分支，一个合并冲突就出现了，接下来我们需要手动去修改它。

$ vim runoob.php

$ cat runoob.php<?php

echo 1;

echo 'runoob';?>

$ git diff

diff --cc runoob.php

index ac60739,b63d7d7..0000000--- a/runoob.php+++ b/runoob.php@@@ -1,3 -1,3 +1,4 @@@

<?php

+echo 1;+ echo 'runoob';

?>

在 Git 中，我们可以用 git add 要告诉 Git 文件冲突已经解决

$ git status -s

UU runoob.php

$ git add runoob.php

$ git status -s

M runoob.php

$ git commit[master 88afe0e] Merge branch 'change\_site'

现在我们成功解决了合并中的冲突，并提交了结果。