创建并切换分支

方式1:

```
➜  test git:(master) git branch dev
➜  test git:(master) git checkout dev
Switched to branch 'dev'
```

方式2:

```
➜  test git:(master) git checkout -b dev
Switched to a new branch 'dev'
```

参数`-b`表示创建并切换，相当于上述的两个指令。

我们可以使用`git branch`查看当前的分支，`git branch`命令会列出所有分支，当前分支前面会标一个`*`号。

```
➜  test git:(dev) git branch
* dev
  master
```

合并分支

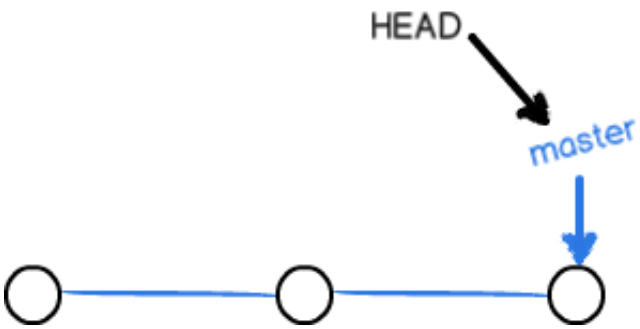
现在我们在`dev`分支上工作，dev分支上只有一个a.txt文件，现在我们新增一个b.txt文件，然后提交到仓库。

```
➜  test git:(dev) ls
a.txt
➜  test git:(dev) touch b.txt
➜  test git:(dev) ✗ git add *
➜  test git:(dev) ✗ git commit -m "新增文件"
```

现在我们切换回`master`分支进行查看，发现刚才新增的文件不见了。

```
➜  test git:(dev) git checkout master
Switched to branch 'master'
➜  test git:(master) ls
a.txt
```

这是因为我们提交的内容还在dev分支上，我们需要把dev分支的工作成果合并到master分支上：




```
➜  test git:(master) git merge dev
Updating 5274c01..cb0e4b8
Fast-forward
 b.txt | 0
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 b.txt
➜  test git:(master) ls
a.txt b.txt
```

git merge命令用于合并指定分支到当前分支。合并后，我们可以看到master分支的内容和dev分支的最新提交是完全一样的。

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。当然，也不是每次合并都能Fast-forward，我们后面会讲其他方式的合并。

合并完成后，就可以放心地删除dev分支了：

```
➜  test git:(master) git branch -d dev
Deleted branch dev (was cb0e4b8).
➜  test git:(master) git branch
```

```
  * master
```

删除之后，我们使用 `git branch` 查看所有分支，发现现在只有master分支了。

因为创建、合并和删除分支非常快，所以Git鼓励你使用分支完成某个任务，合并后再删掉分支，这和直接在master分支上工作效果是一样的，但过程更安全。