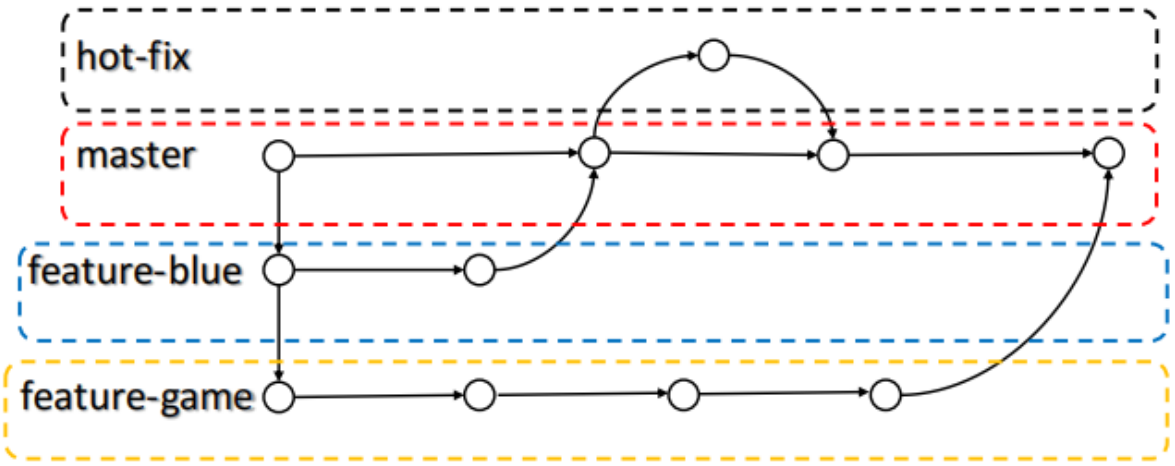
第4讲 Git 分支

什么是分支

在版本控制过程中，同时推进多个任务，为每个任务，我们就可以创建每个任务的单独分支。使用分支意味着程序员可以把自己的工作从开发主线上分离开来，开发自己分支的时候，不会影响主线分支的运行。对于初学者而言，分支可以简单理解为副本，一个分支就是一个单独的副本。（分支底层其实也是指针的引用）



分支的好处

同时并行推进多个功能开发，提高开发效率。

各个分支在开发过程中，如果某一个分支开发失败，不会对其他分支有任何影响。失败的分支删除重新开始即可。

分支的操作

命令名称	作用
`git branch`	分支名 创建分支
`git branch -v`	查看分支
`git checkout`	分支名 切换分支
`git merge`	分支名 把指定的分支合并到当前分支上

查看分支

- 基本语法

```
git branch -v
```

```
hp@LAPTOP-ARLL3DSO MINGW64 /e/Git-Space/git-test (master)
$ git branch -v
* master 5228695 第三次提交
```

创建分支

- `git branch` 分支名

```
hp@LAPTOP-ARLL3DSO MINGW64 /e/Git-Space/git-test (master)
$ git branch hot-fix

hp@LAPTOP-ARLL3DSO MINGW64 /e/Git-Space/git-test (master)
$ git branch -v
  hot-fix 5228695 第三次提交
* master  5228695 第三次提交
```

切换分支

- 基本语法

```
git checkout 分支名
```

1. 切换分支

```
hp@LAPTOP-ARLL3DSO MINGW64 /e/Git-Space/git-test (master)
$ git checkout hot-fix
Switched to branch 'hot-fix'
```

2. 查看分支

```
hp@LAPTOP-ARLL3DSO MINGW64 /e/Git-Space/git-test (hot-fix)
$ git branch -v
* hot-fix 5228695 第三次提交
  master  5228695 第三次提交
```

3. 修改文件

```
hp@LAPTOP-ARLL3DSO MINGW64 /e/Git-Space/git-test (hot-fix)
$ vim test.txt
```

4. 查看状态

```
hp@LAPTOP-ARLL3DSO MINGW64 /e/Git-Space/git-test (hot-fix)
$ git status
On branch hot-fix
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   test.txt

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

5. 添加暂存区

```
hp@LAPTOP-ARLL3DSO MINGW64 /e/Git-Space/git-test (hot-fix)
$ git add test.txt
```

6. 提交本地库

```
hp@LAPTOP-ARLL3DSO MINGW64 /e/Git-Space/git-test (hot-fix)
$ git commit -m "hot-fix 第一次提交" test.txt
[hot-fix f5811dd] hot-fix 第一次提交
 1 file changed, 1 insertion(+)
```

7. 查看文本

```
hp@LAPTOP-ARLL3DSO MINGW64 /e/Git-Space/git-test (hot-fix)
$ cat test.txt
yxyxyxyxyx
xyxyyxyxyxyxyx
yx
hot-fix
```

8. 查看日志

```
hp@LAPTOP-ARLL3DSO MINGW64 /e/Git-Space/git-test (hot-fix)
$ git reflog
f5811dd (HEAD -> hot-fix) HEAD@{0}: commit: hot-fix 第一次提交
5228695 (master) HEAD@{1}: checkout: moving from master to hot-fix
5228695 (master) HEAD@{2}: reset: moving to 5228695
c5bbd23 HEAD@{3}: reset: moving to c5bbd23
985076e HEAD@{4}: reset: moving to 985076e
5228695 (master) HEAD@{5}: commit: 第三次提交
985076e HEAD@{6}: commit: 第二次提交
c5bbd23 HEAD@{7}: commit (initial): 第一次提交
```

合并分支

- 基本语法

```
git merge 分支名
```

正常合并分支

1. 切换回 `master` 分支

```
hp@LAPTOP-ARLL3DSO MINGW64 /e/Git-Space/git-test (hot-fix)
$ git checkout master
Switched to branch 'master'
```

2. 查看文件

```
hp@LAPTOP-ARLL3DSO MINGW64 /e/Git-Space/git-test (master)
$ cat test.txt
yxyxyxyxyx
xyxyyxyxyxyxyx
yx
```

3. 在 `master` 分支上合并 `hot-fix` 分支

```
hp@LAPTOP-ARLL3DSO MINGW64 /e/Git-Space/git-test (master)
$ git merge hot-fix
Updating 5228695..f5811dd
Fast-forward
 test.txt | 1 +
 1 file changed, 1 insertion(+)
```

4. 再次查看合并后文件。

```
hp@LAPTOP-ARLL3DSO MINGW64 /e/Git-Space/git-test (master)
$ cat test.txt
yxyxyxyxyx
xyxyyxyxyxyxyx
yx
hot-fix
```

产生冲突

冲突产生的原因： 合并分支时，两个分支在同一个文件的同一个位置有两套完全不同的修改。`Git` 无法替 我们决定使用哪一个。必须人为决定新代码内容。

1. 修改`master`分支文件

```
hp@LAPTOP-ARLL3DSO MINGW64 /e/Git-Space/git-test (master)
$ vim test.txt
```

2. 添加暂存区，提交本地库

```
hp@LAPTOP-ARLL3DSO MINGW64 /e/Git-Space/git-test (master)
$ git add test.txt

hp@LAPTOP-ARLL3DSO MINGW64 /e/Git-Space/git-test (master)
$ git commit -m "master 第四次提交" test.txt
[master ce1bbf5] master 第四次提交
 1 file changed, 2 insertions(+)
```

3. 查看文件

```
hp@LAPTOP-ARLL3DSO MINGW64 /e/Git-Space/git-test (master)
$ cat test.txt
yxyxyxyxyx
```

```
xyxyyxyxyxyxyx
yx
hot-fix
master1
```

4. 切换`hot-fix`分支

```
hp@LAPTOP-ARLL3DSO MINGW64 /e/Git-Space/git-test (master)
$ git checkout hot-fix
Switched to branch 'hot-fix'
```

5. 修改文件

```
hp@LAPTOP-ARLL3DSO MINGW64 /e/Git-Space/git-test (hot-fix)
$ vim test.txt
```

6. 添加暂存区，提交本地库

```
hp@LAPTOP-ARLL3DSO MINGW64 /e/Git-Space/git-test (hot-fix)
$ git add test.txt

hp@LAPTOP-ARLL3DSO MINGW64 /e/Git-Space/git-test (hot-fix)
$ git commit -m "hotfix 提交" test.txt
[hot-fix 9cfa1d8] hotfix 提交
 1 file changed, 1 insertion(+), 1 deletion(-)
```

7. 切换回`master`分支

```
hp@LAPTOP-ARLL3DSO MINGW64 /e/Git-Space/git-test (hot-fix)
$ git checkout master
Switched to branch 'master'
```

8. 合并`hot-fix`分支

```
hp@LAPTOP-ARLL3DSO MINGW64 /e/Git-Space/git-test (master)
$ git merge hot-fix
Auto-merging test.txt
CONFLICT (content): Merge conflict in test.txt
Automatic merge failed; fix conflicts and then commit the result.
```

9. 查看状态

- 冲突产生的表现：后面状态为`MERGING`

```
hp@LAPTOP-ARLL3DSO MINGW64 /e/Git-Space/git-test (master|MERGING)
$ git status
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:   test.txt

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

10. 修改文件

- 编辑有冲突的文件，删除特殊符号，决定要使用的内容
- 特殊符号：<<<<<<< HEAD 当前分支的代码 ======= 合并过来的代码 >>>>>>> hot-fix

```
hp@LAPTOP-ARLL3DSO MINGW64 /e/Git-Space/git-test (master|MERGING)
$ vim test.txt

hp@LAPTOP-ARLL3DSO MINGW64 /e/Git-Space/git-test (master|MERGING)
$ git status
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:   test.txt

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

11. 添加暂存区，提交本地库

- 执行提交（注意：此时使用 `git commit` 命令时不能带文件名）

```
hp@LAPTOP-ARLL3DSO MINGW64 /e/Git-Space/git-test (master|MERGING)
$ git add test.txt

hp@LAPTOP-ARLL3DSO MINGW64 /e/Git-Space/git-test (master|MERGING)
$ git commit -m "merge" test.txt
fatal: cannot do a partial commit during a merge.

// 发现后面 MERGING 消失，变为正常

hp@LAPTOP-ARLL3DSO MINGW64 /e/Git-Space/git-test (master|MERGING)
$ git commit -m "merge"
[master 6ad1431] merge
```

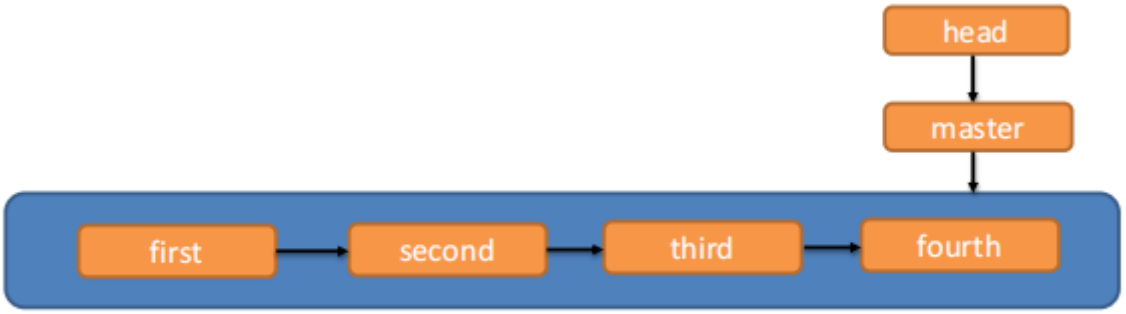
12. 查看文件

```
hp@LAPTOP-ARLL3DSO MINGW64 /e/Git-Space/git-test (master)
$ cat text.txt
cat: text.txt: No such file or directory

hp@LAPTOP-ARLL3DSO MINGW64 /e/Git-Space/git-test (master)
$ cat test.txt
yxyxyxyxyx
xyxyyxyxyxyxyx
yx
master1
hot-fix1
```

13. 切换回 `hot-fix` 分支查看文件

```
hp@LAPTOP-ARLL3DSO MINGW64 /e/Git-Space/git-test (master)
$ git checkout hot-fix
Switched to branch 'hot-fix'

hp@LAPTOP-ARLL3DSO MINGW64 /e/Git-Space/git-test (hot-fix)
$ cat test.txt
yxyxyxyxyx
xyxyyxyxyxyxyx
yx
hot-fix1
```

创建分支和切换分支图解



`master`、`hot-fix` 其实都是指向具体版本记录的指针。当前所在的分支，其实是由 `HEAD` 决定的。所以创建分支的本质就是多创建一个指针。

`HEAD` 如果指向 `master`，那么我们现在就在 `master` 分支上。

`HEAD` 如果执行 `hotfix`，那么我们现在就在 `hotfix` 分支上

所以切换分支的本质就是移动 `HEAD` 指针