

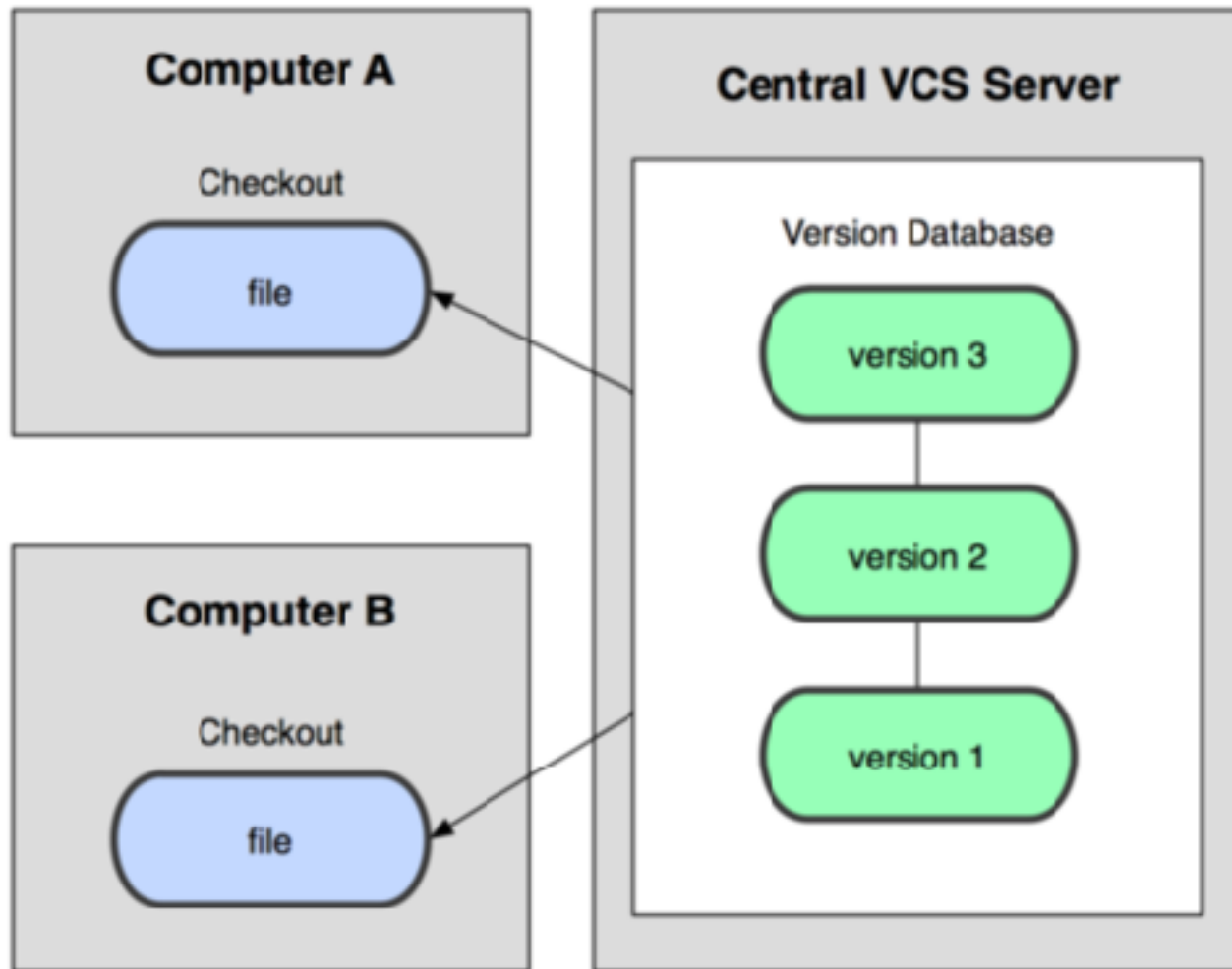


git

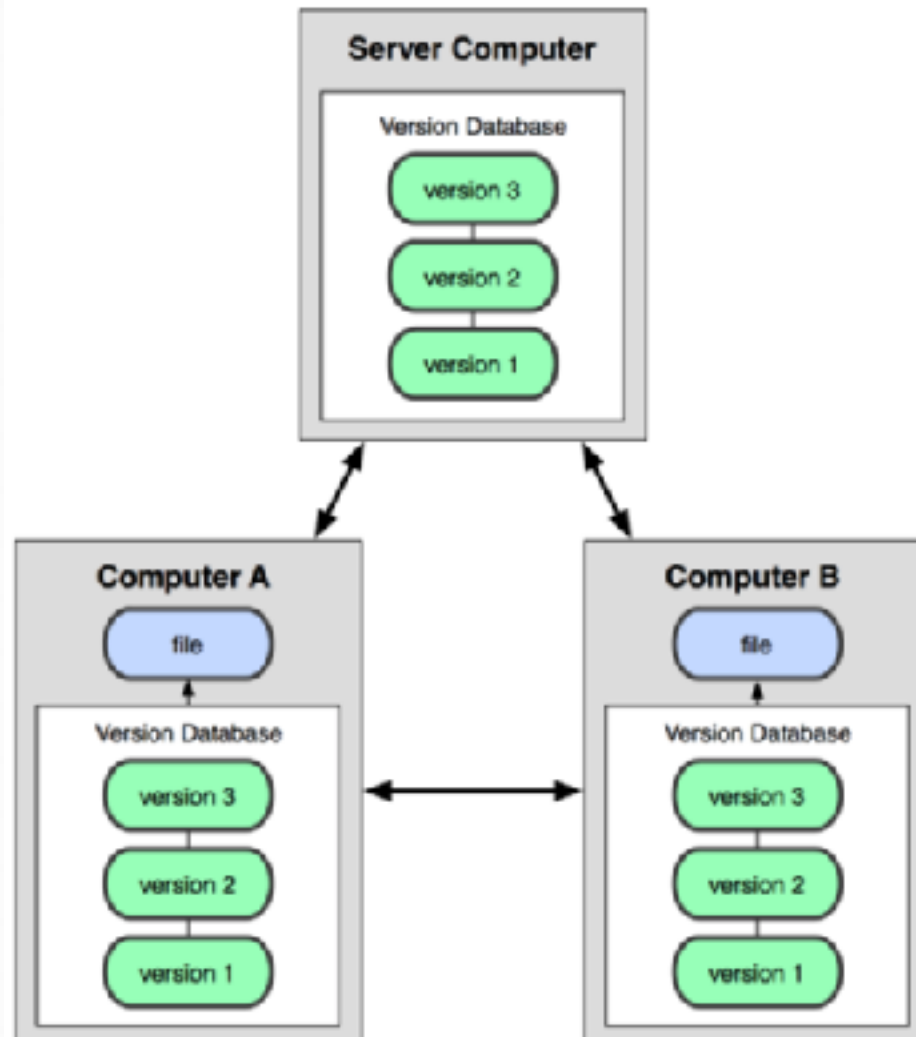
大纲

- ① 关于版本控制
- ② git基础
- ③ git分支
- ④ gitflow工作流

集中化的版本控制系统



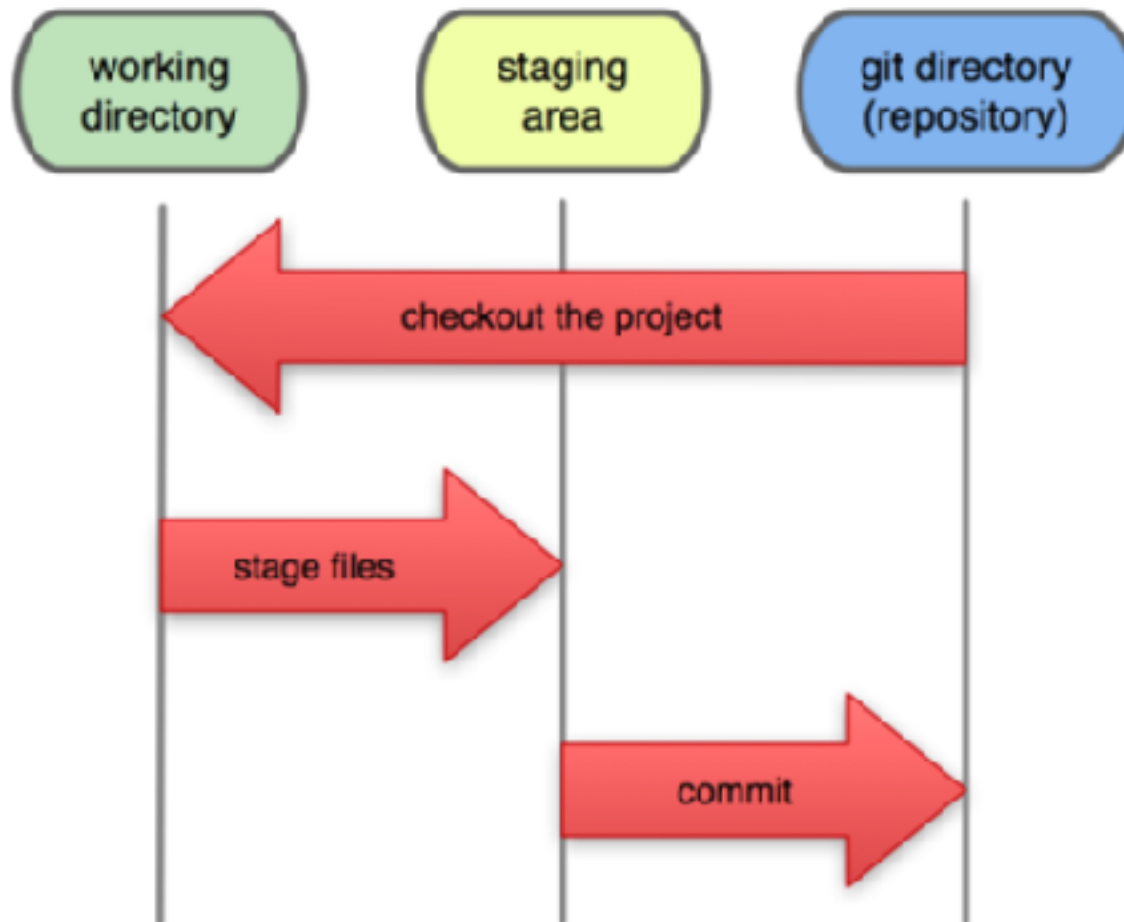
分布式版本控制系统



git的基础要点

- 直接快照，而非比较差异
- 近乎所有操作都可本地执行
- 时刻保持数据完整性
- 多数操作仅添加数据
- 三种状态

Local Operations

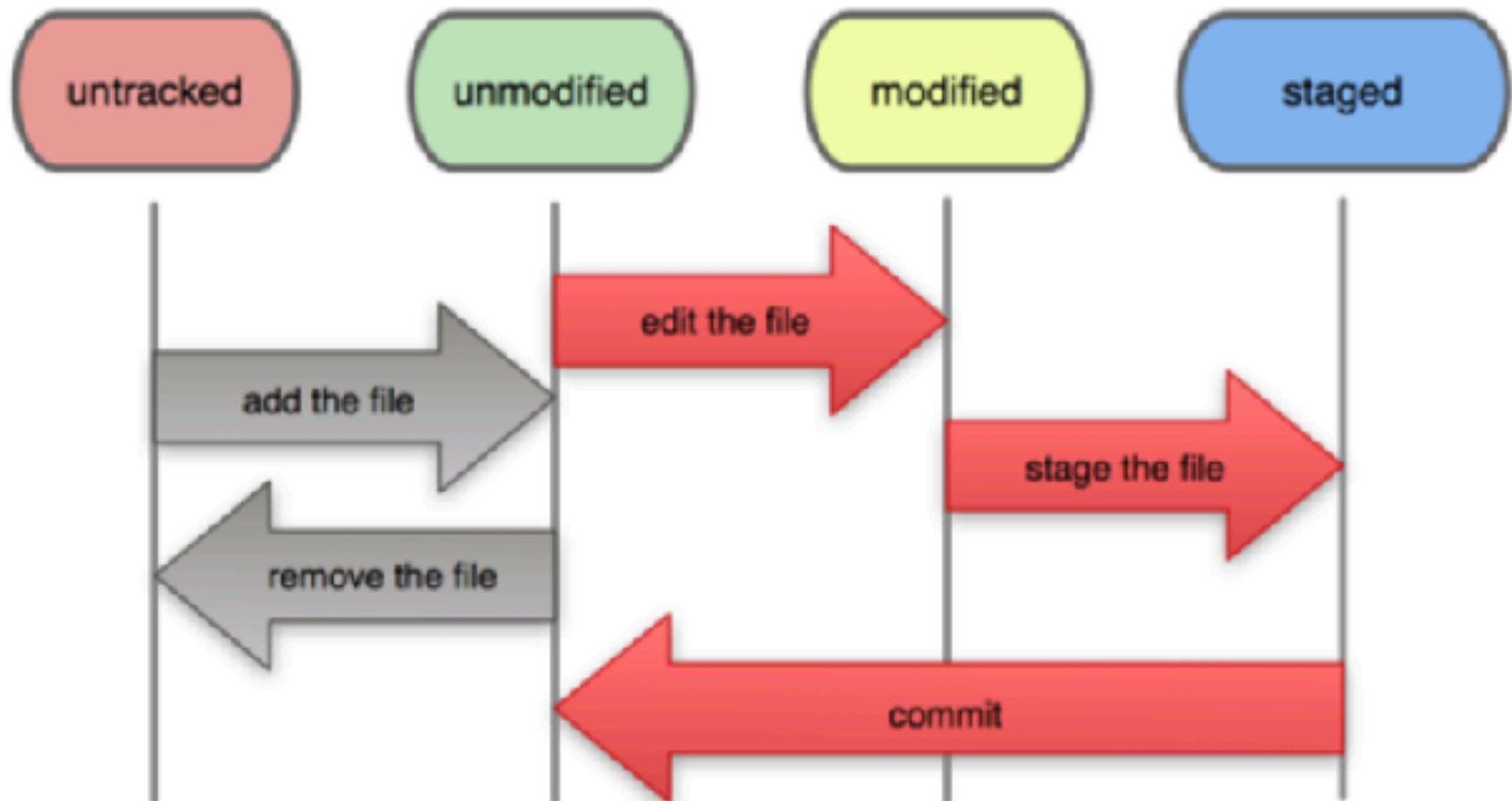


工作目录，暂存区域和git目录

取得项目的git仓库

- 从当前目录初始化 `git init`
- 从现有仓库克隆 `git clone`

File Status Lifecycle



记录每次更新到仓库

- 检查当前文件状态 `git status`
- 跟踪新文件 `git add README`
- 暂存已修改的文件 `git add modified.txt`
- 忽略某些文件 `.gitignore`
- 提交更新 `git commit`
- 移除文件 `git rm readme.txt`
- 移动文件 `git mv file_from file_to`

查看提交历史

- `git log`
- `git log -p -2`
- `git log --stat`
- `git log --pretty=format:"%h - %an : %s" --graph`
- `git log --since=2.weeks`

撤消操作

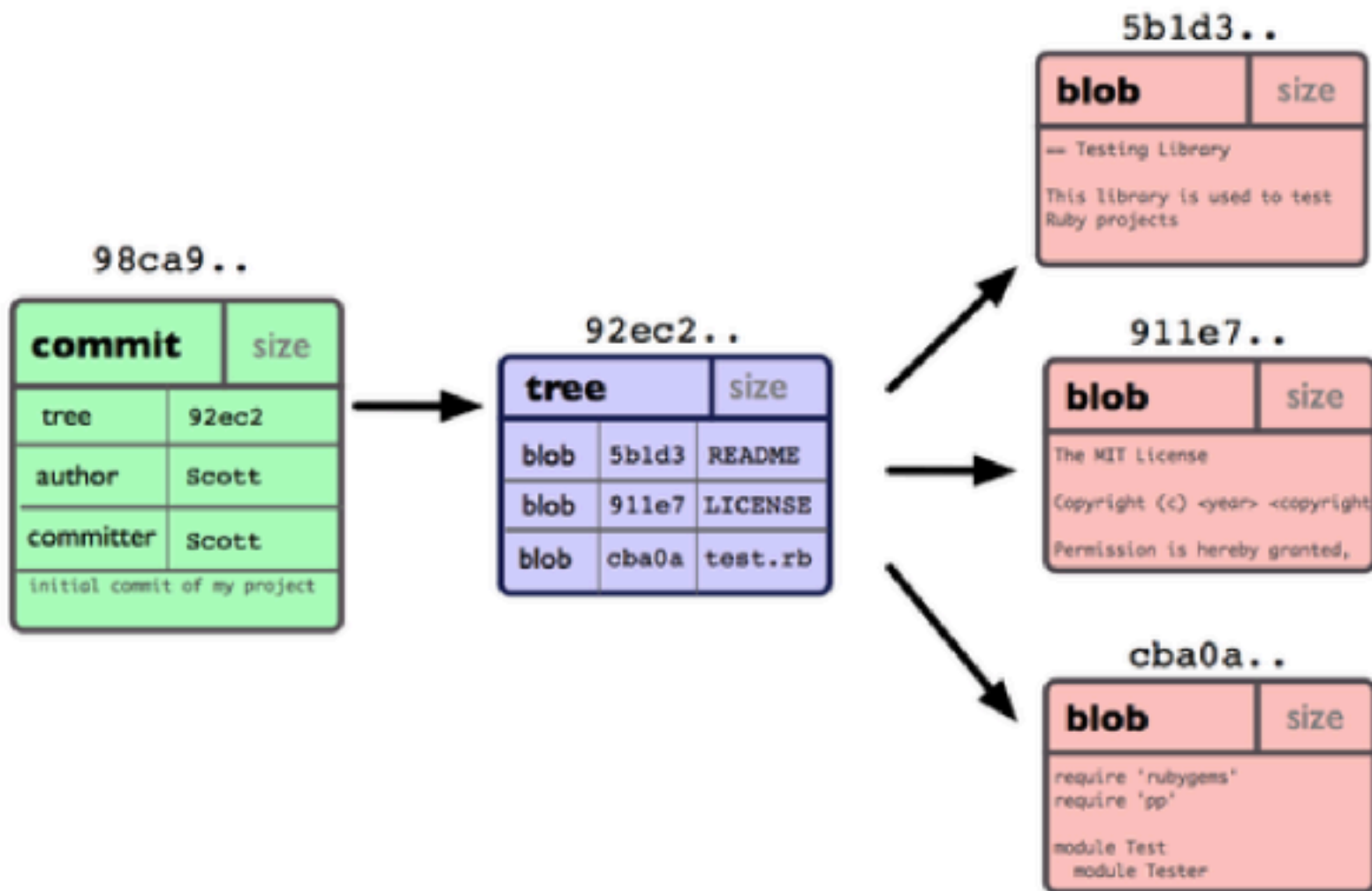
- 修改最后一次提交 `git commit --amend`
- 取消已经暂存的文件 `git reset HEAD readme.txt`
- 取消对文件的修改 `git checkout -- readme.txt`
- `git stash`

远程仓库

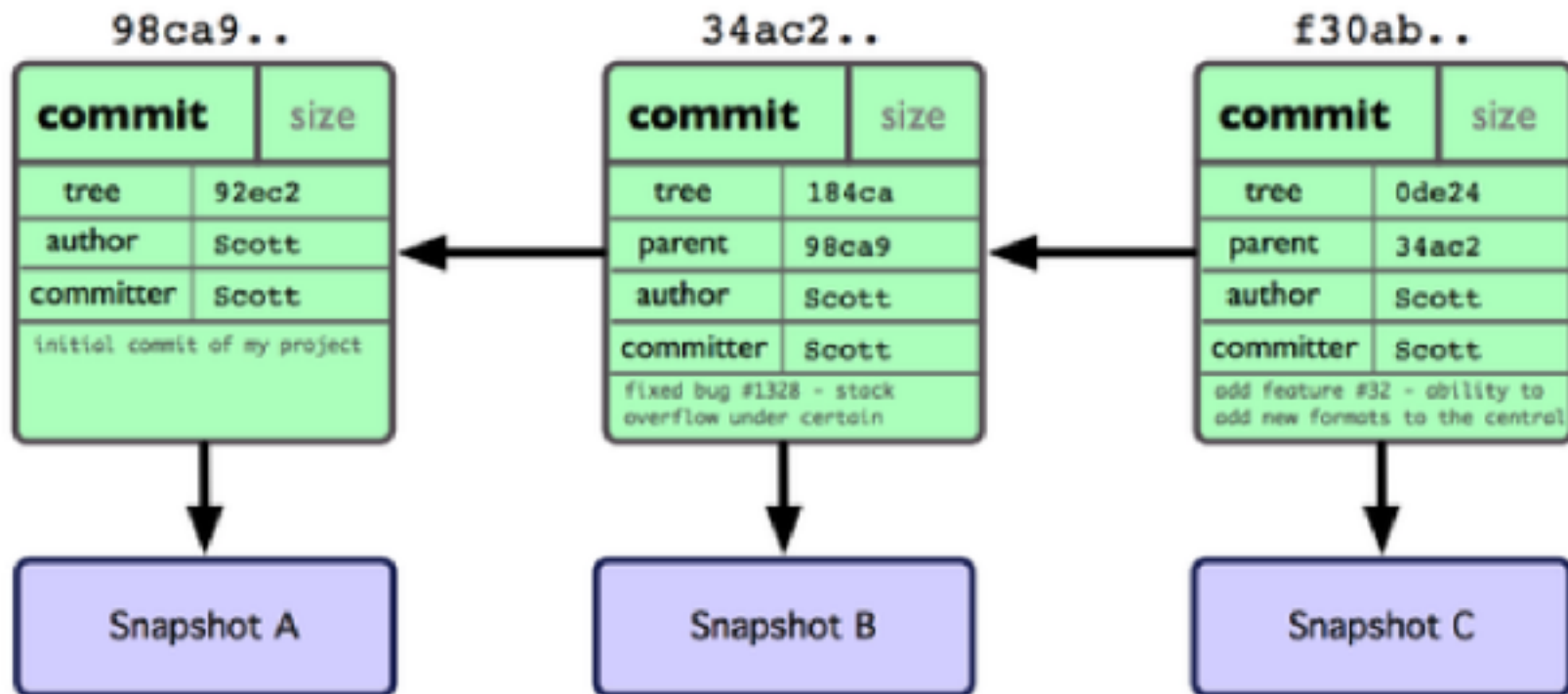
- 查看当前的远程仓库 `git remote -v`
- 添加远程仓库 `git remote add pb git://git.osc/xxx/abc.git`
- 从远程仓库抓取数据 `git fetch[remote-name]`
- 推送数据以远程仓库 `git push origin master`
- 查看远程仓库信息 `git remote show origin`
- 重命名远程仓库 `git remote rename tom jerry`
- 删除远程仓库 `git remote rm jerry`

git分支

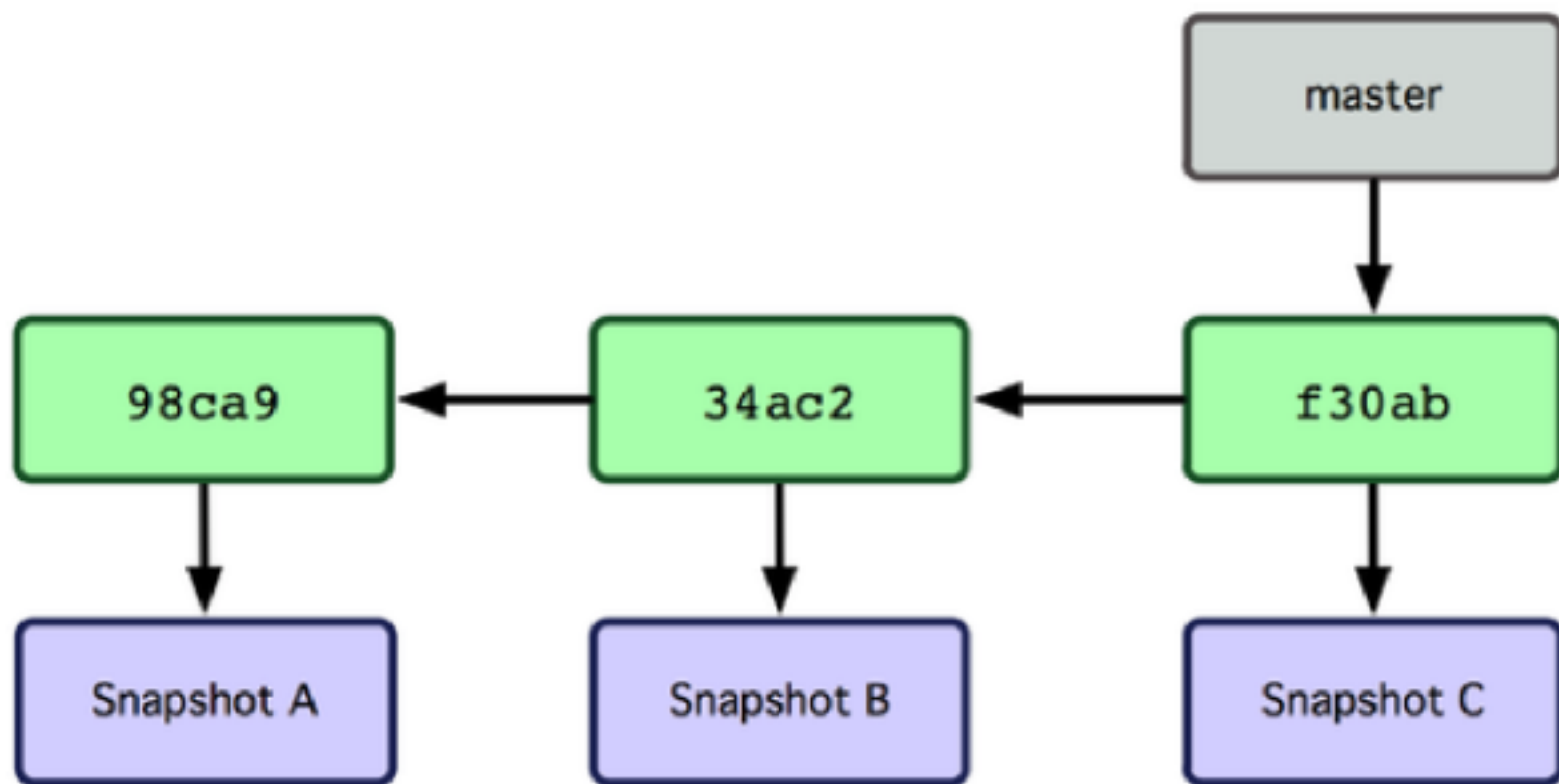
- 轻量
- 瞬间创建
- 本质上仅仅是指向commit对象的可变指针



一次提交后仓库里的数据



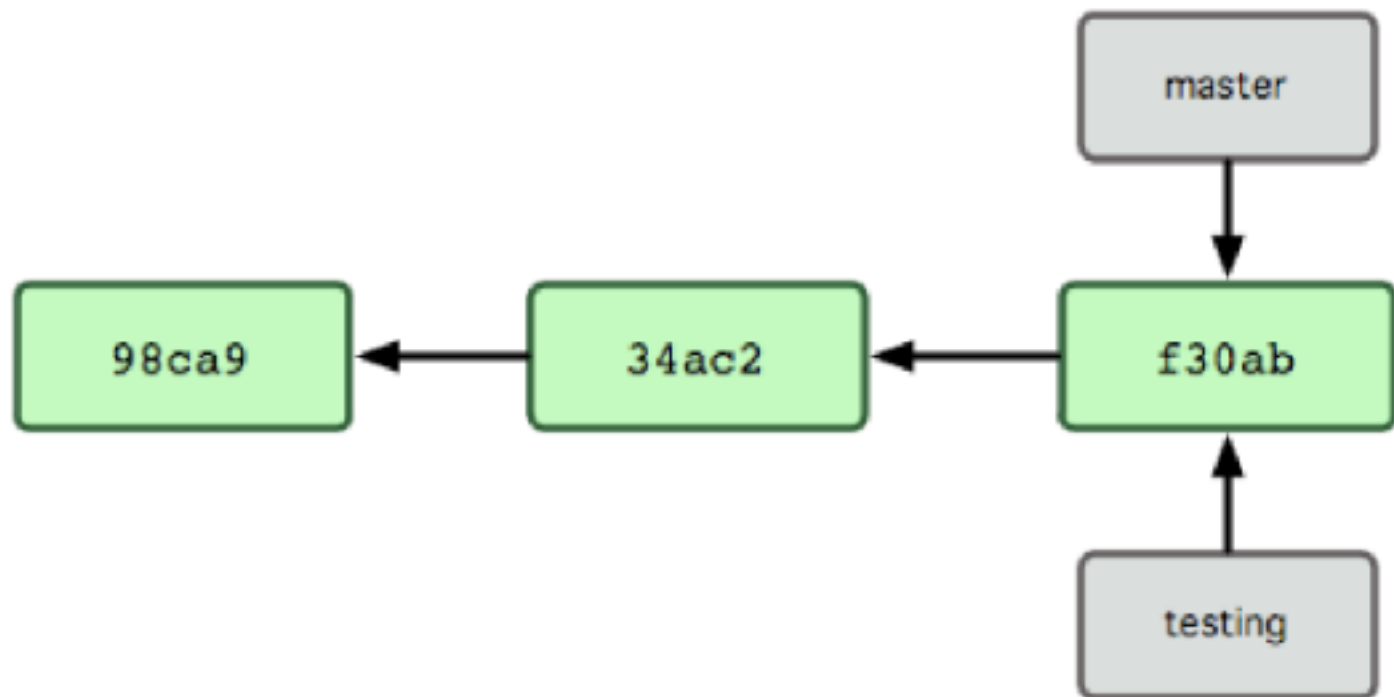
多次提交后的git对象数据



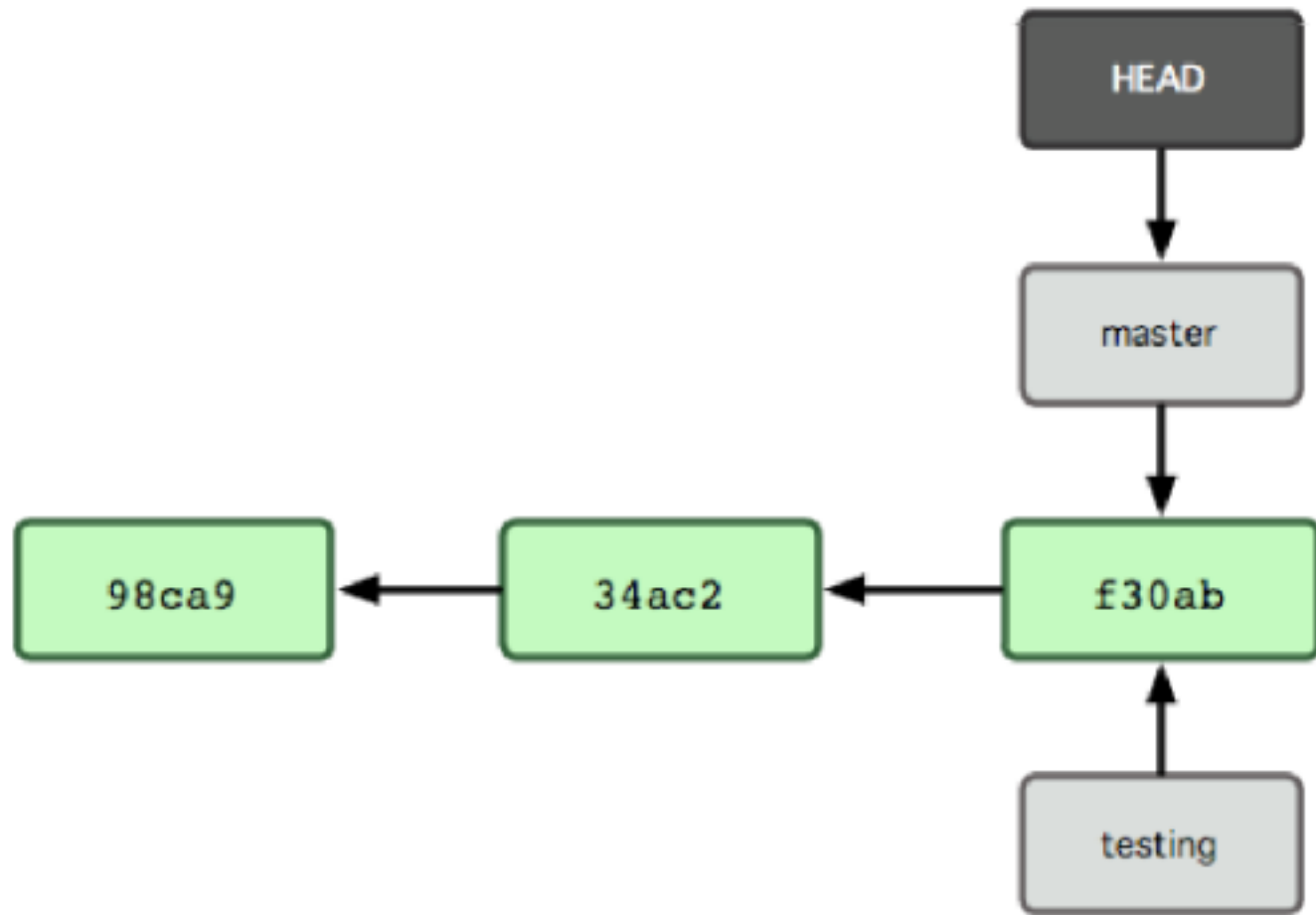
指向提交数据历史的分支

git branch testing

新建分支

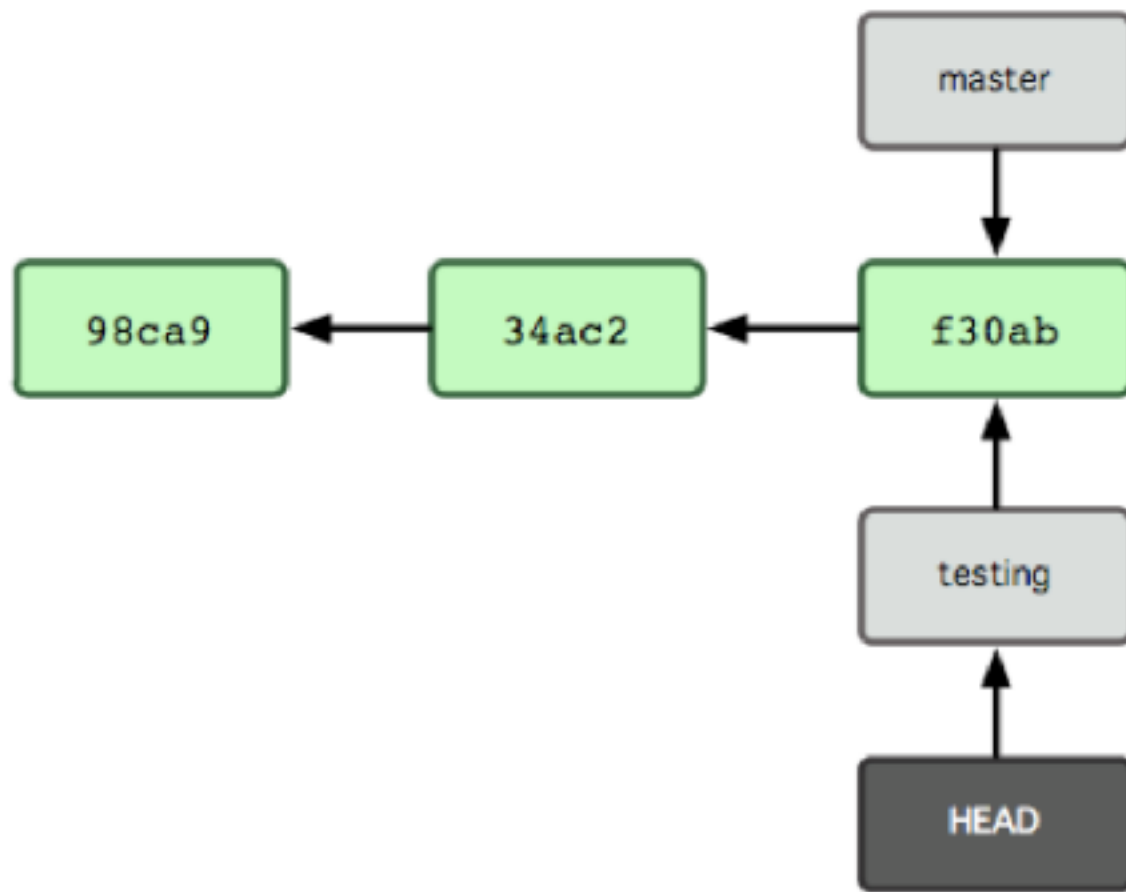


多个分支指向提交数据的历史

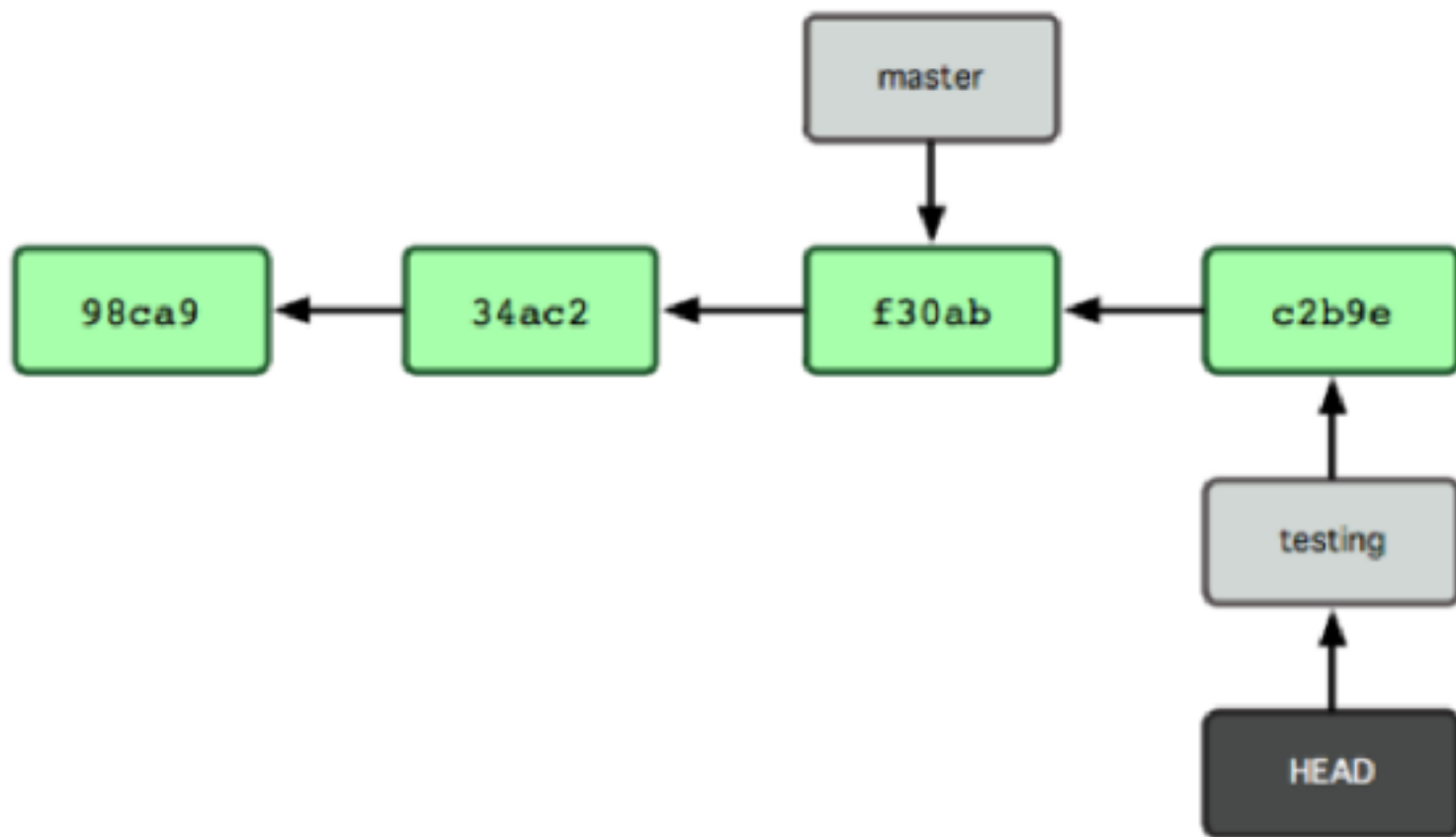


HEAD指向当前所在的分支

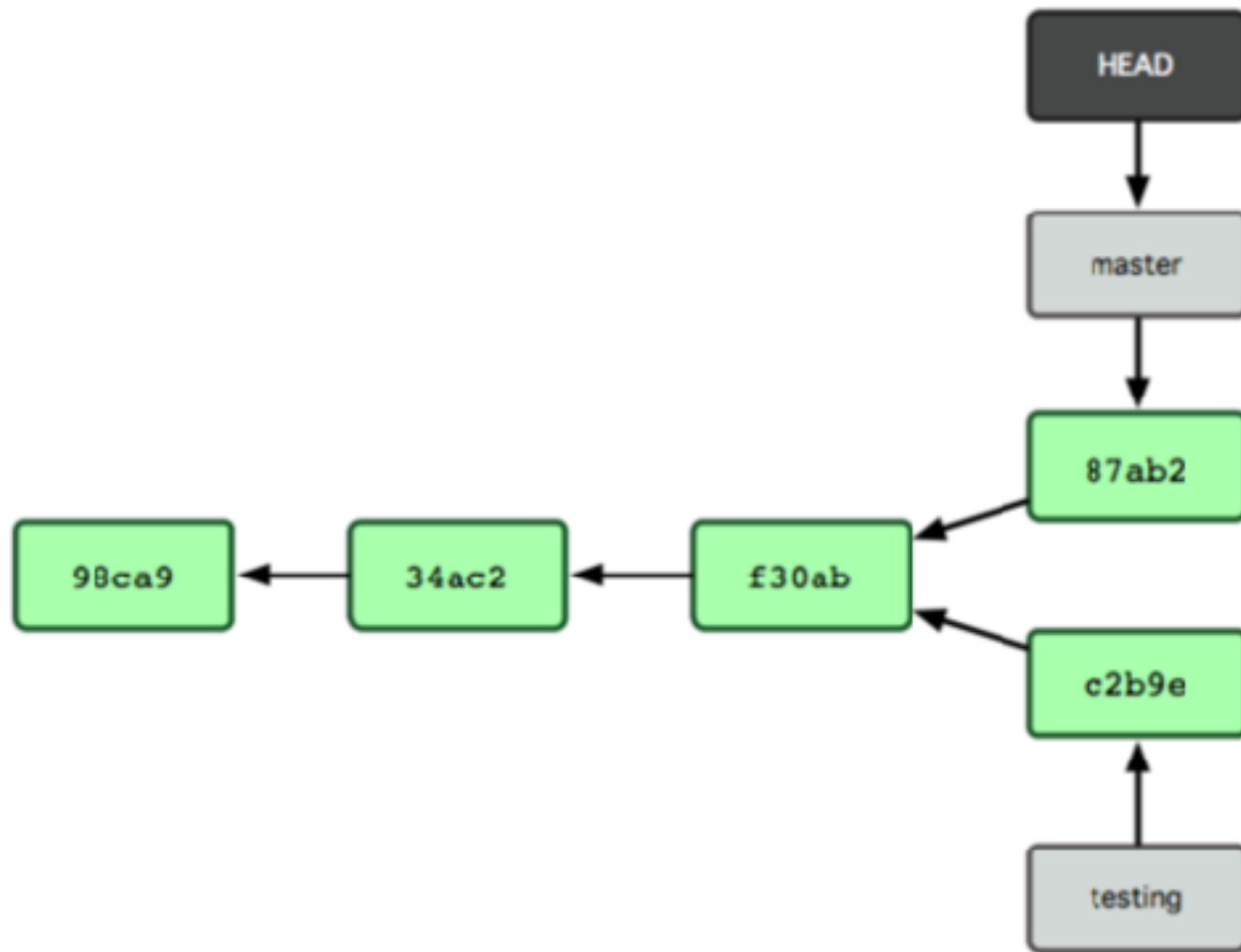
git checkout testing



HEAD在你转换分支时指向新的分支

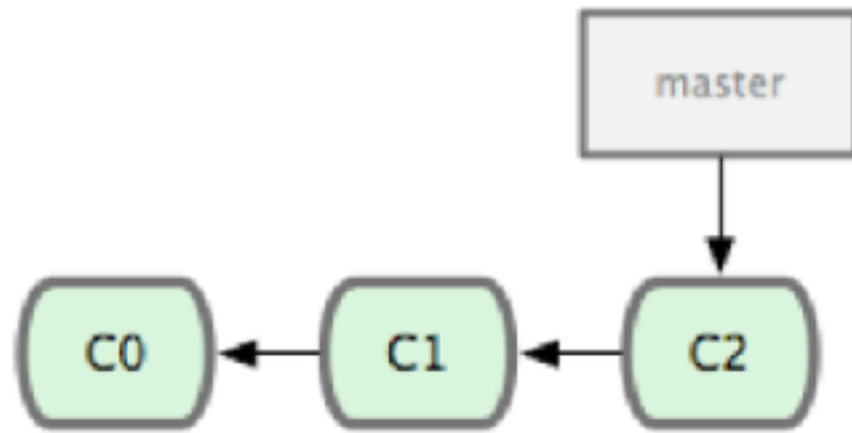


每次提交后HEAD随着分支一起向前移动

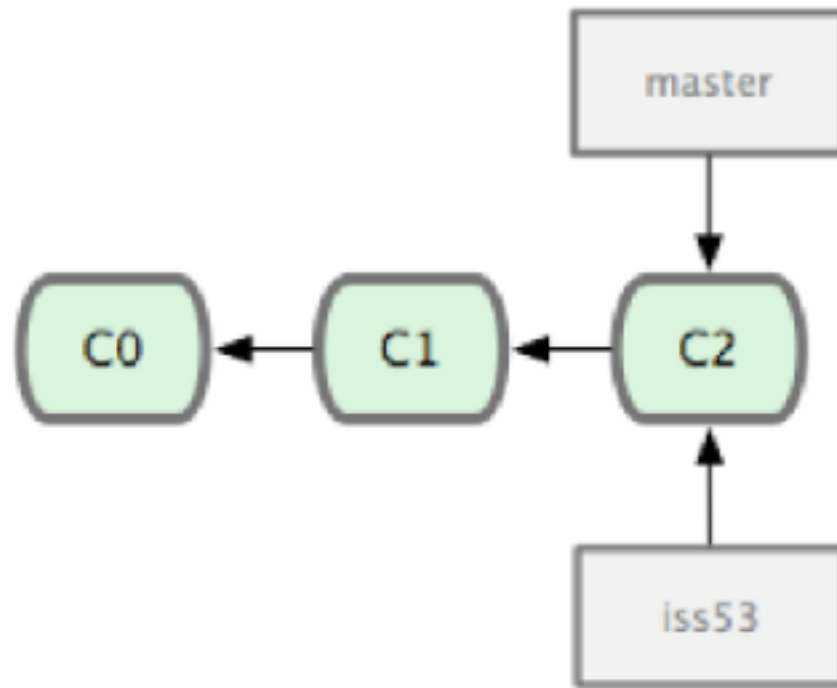


分叉了的分支历史

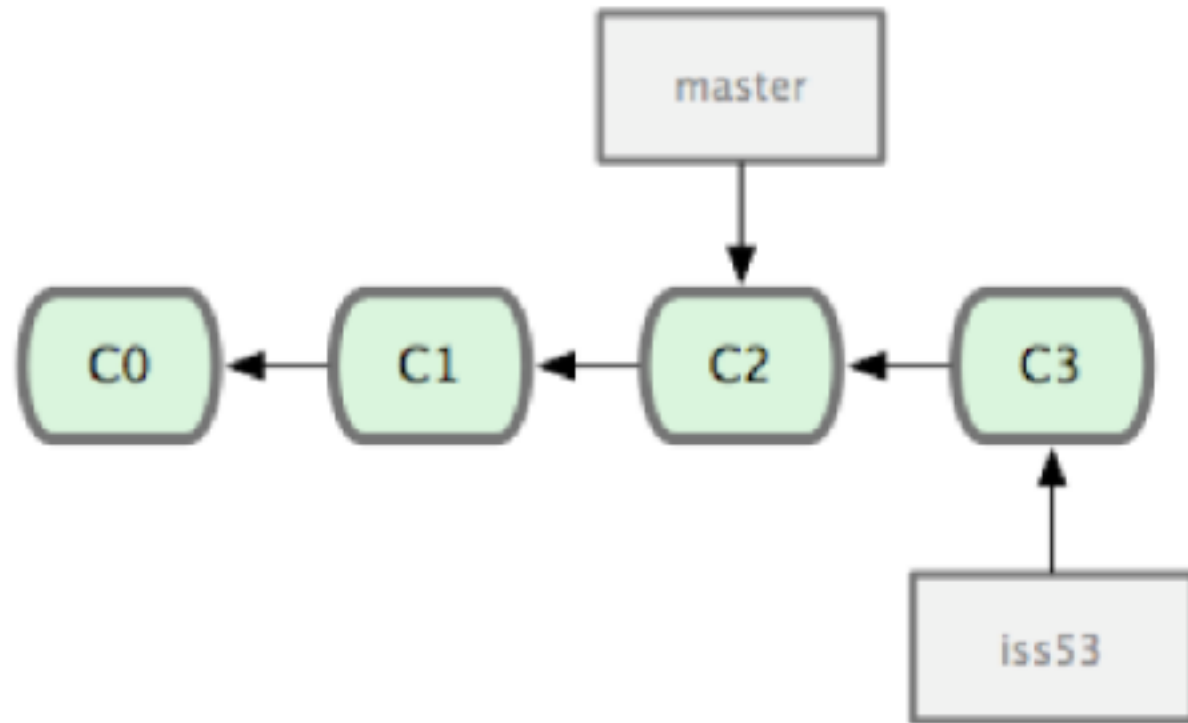
分支与合并



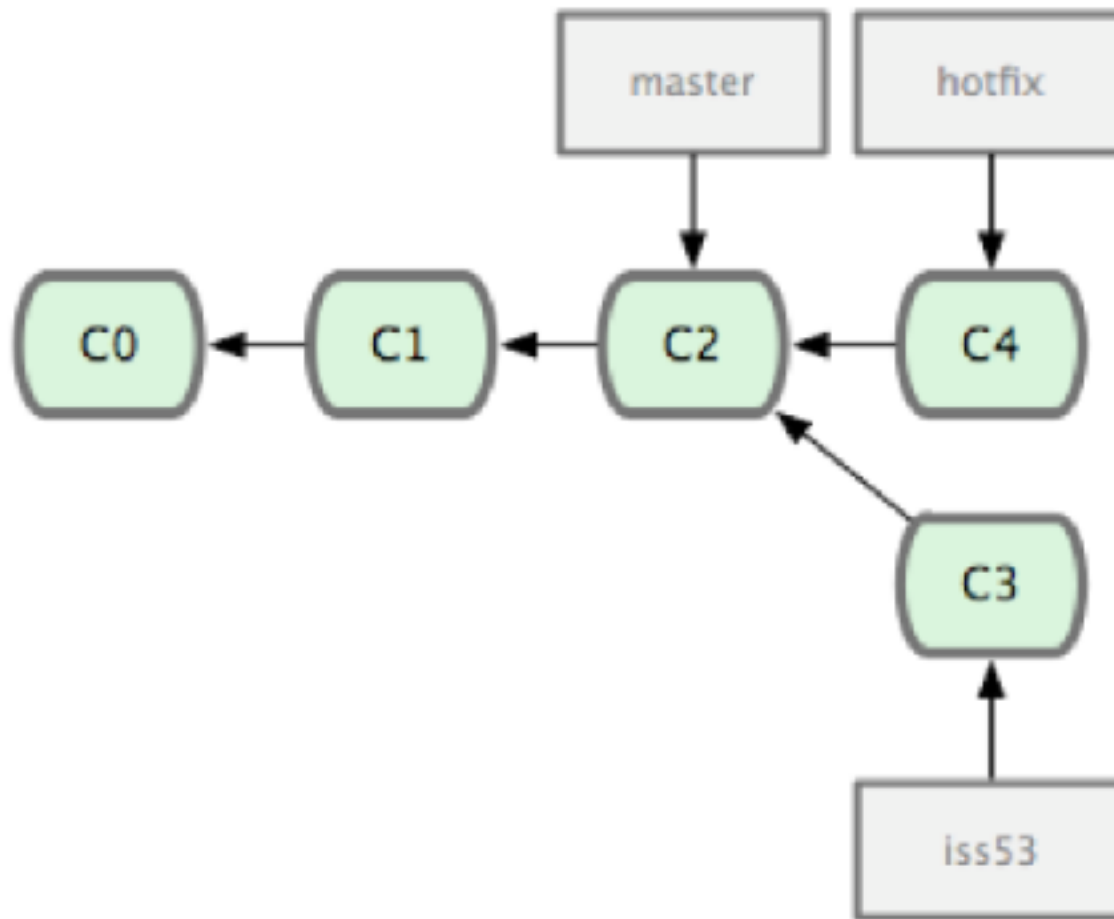
一部分简短的提交历史



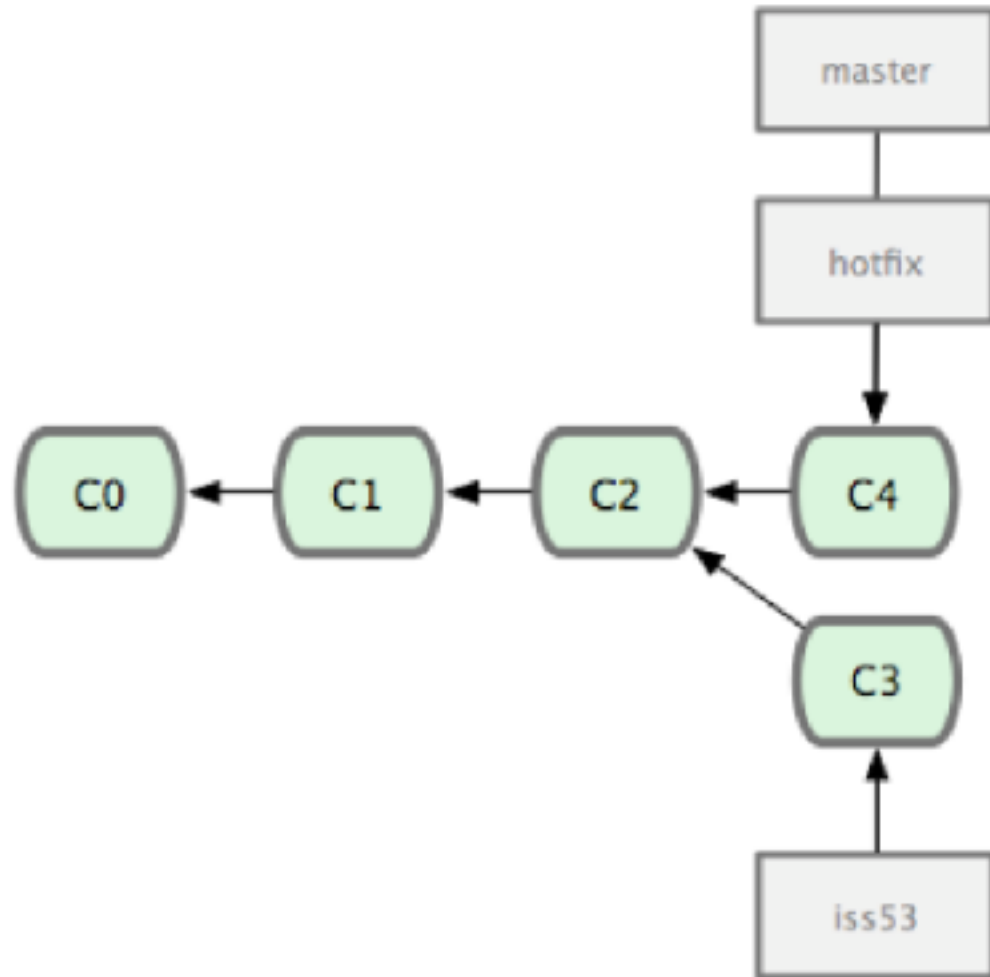
创建了一个新的分支指针



iss53分支随工作进展向前推进

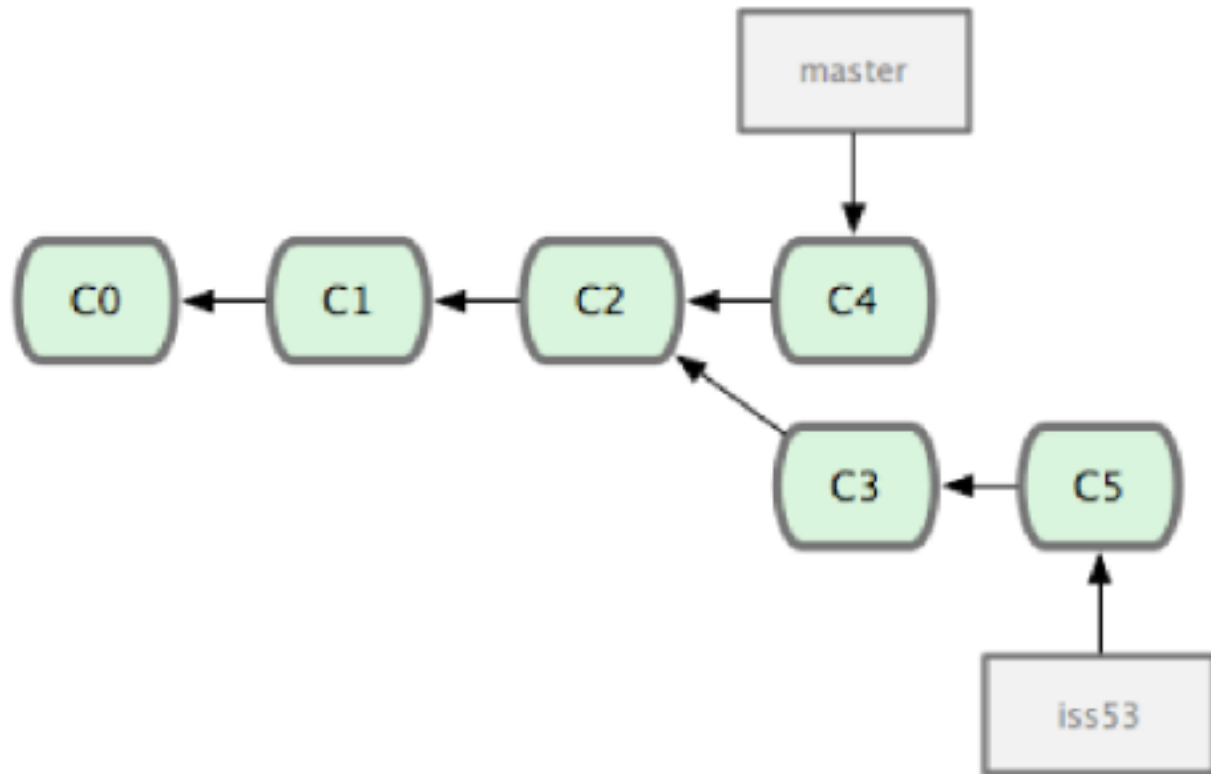


hotfix分支是从master分支所在点分化出来的



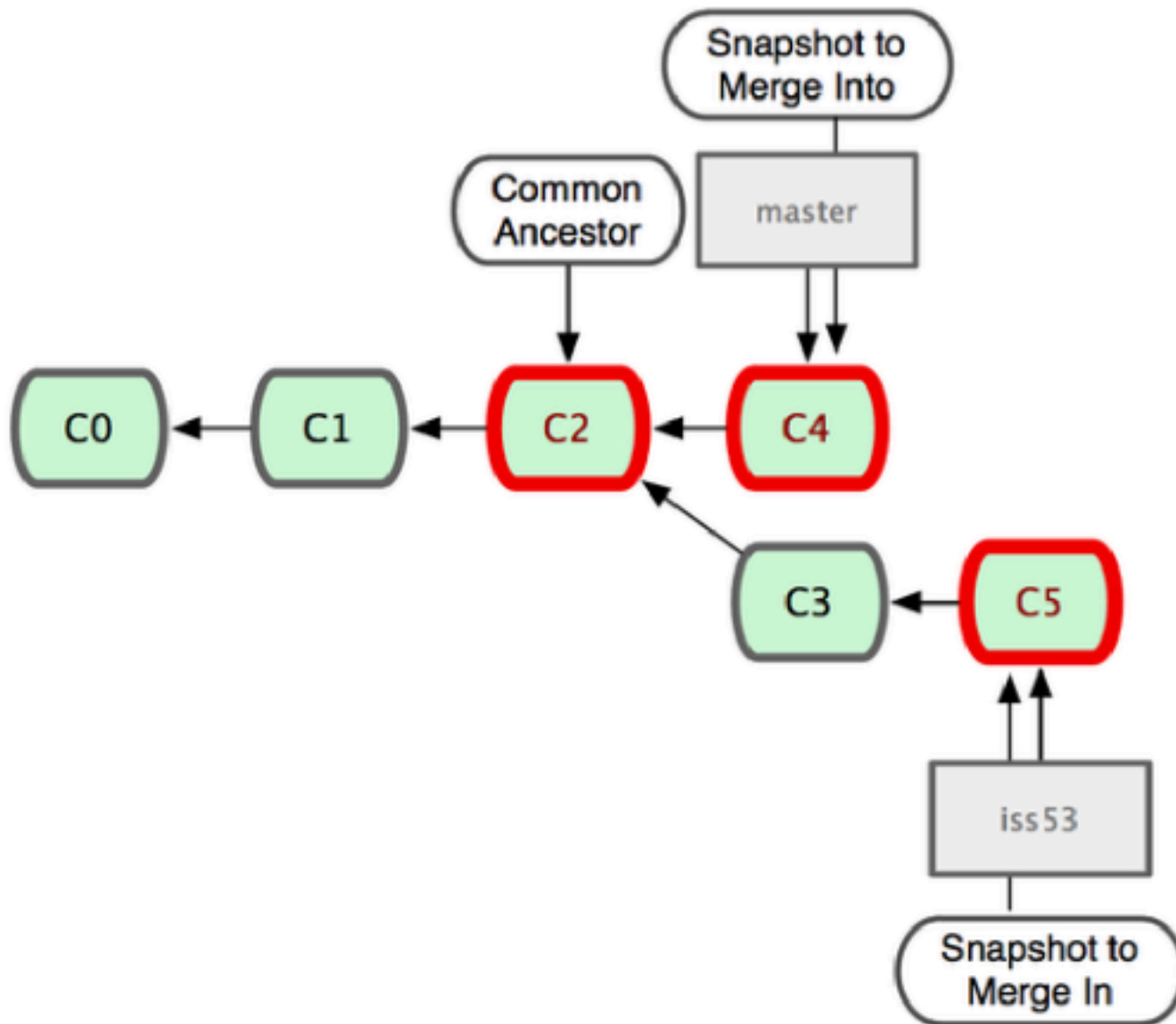
合并之后，master分支和hotfix分支指向同一位置

git branch -d hotfix

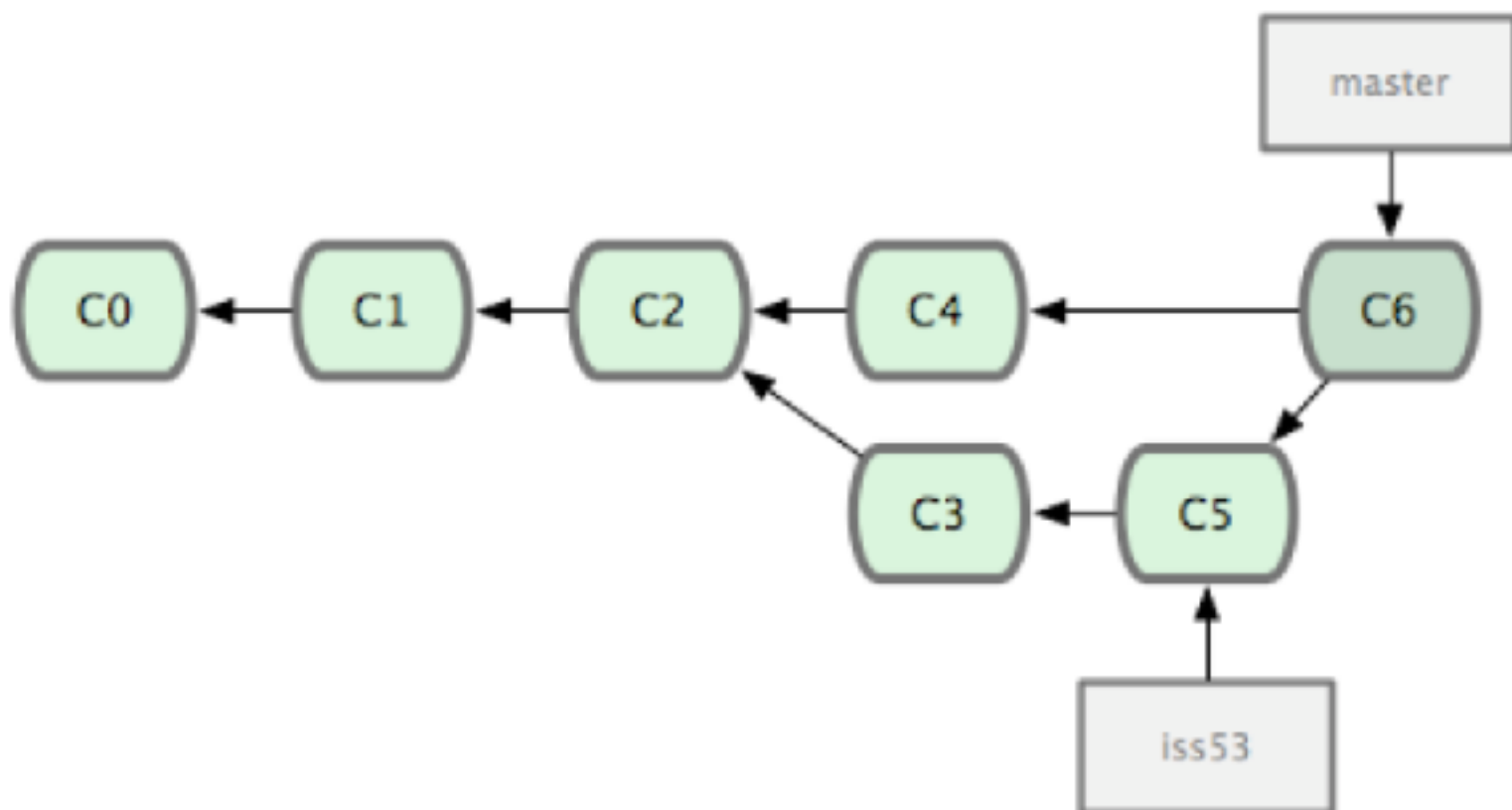


git checkout iss53

iss53分支不受影响继续推进



git为分支合并自动识别出最佳的同源合并点




git自动创建了一个包含了合并结果的commit对象

冲突的合并

```
$ git merge iss53  
Auto-merging index.html  
CONFLICT (content): Merge conflict in index.html  
Automatic merge failed; fix conflicts and then commit the result.
```

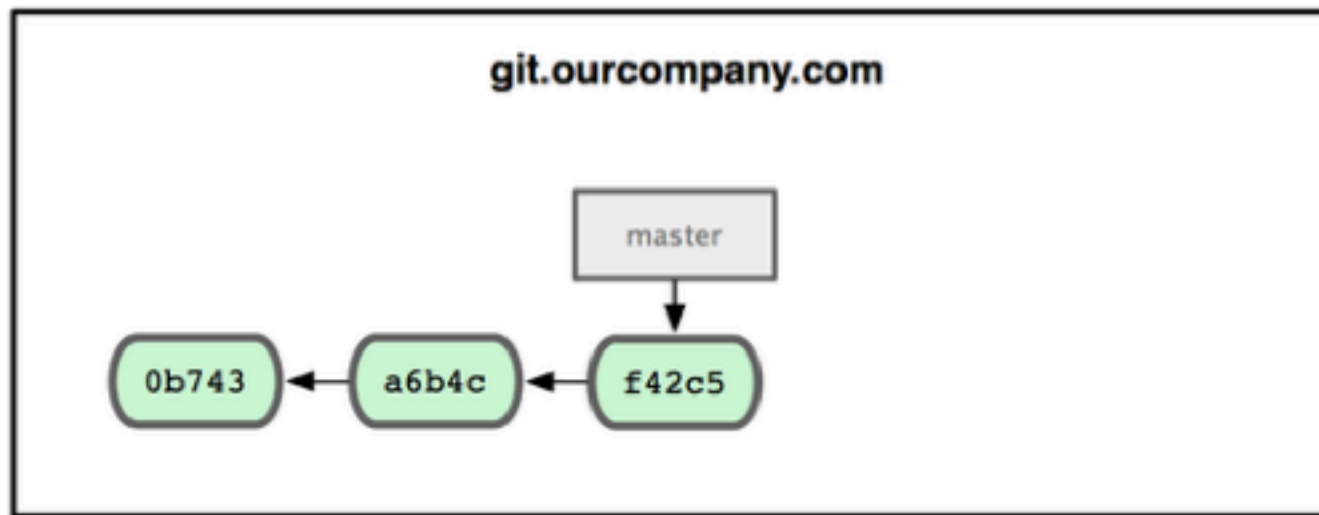
修改了待合并分支里同一个文件的同一部分



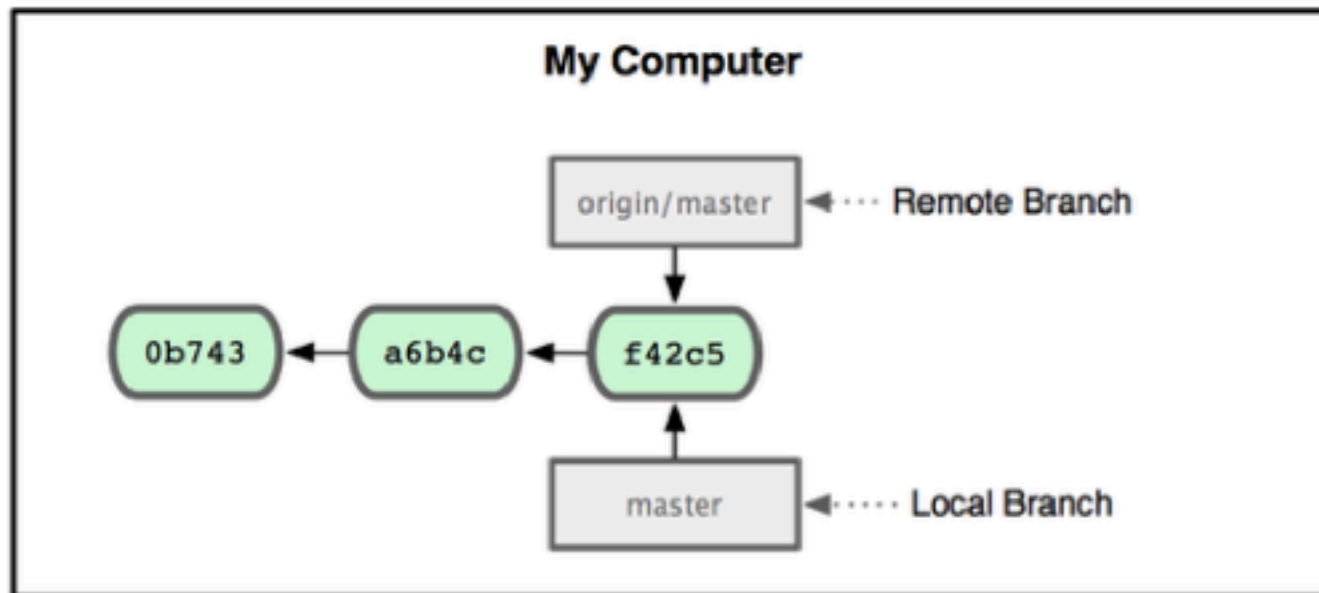
```
<<<<<<< HEAD:index.html
<div id="footer">contact : email.support@github.com</div>
=====
<div id="footer">
  please contact us at support@github.com
</div>
>>>>>>> iss53:index.html
```

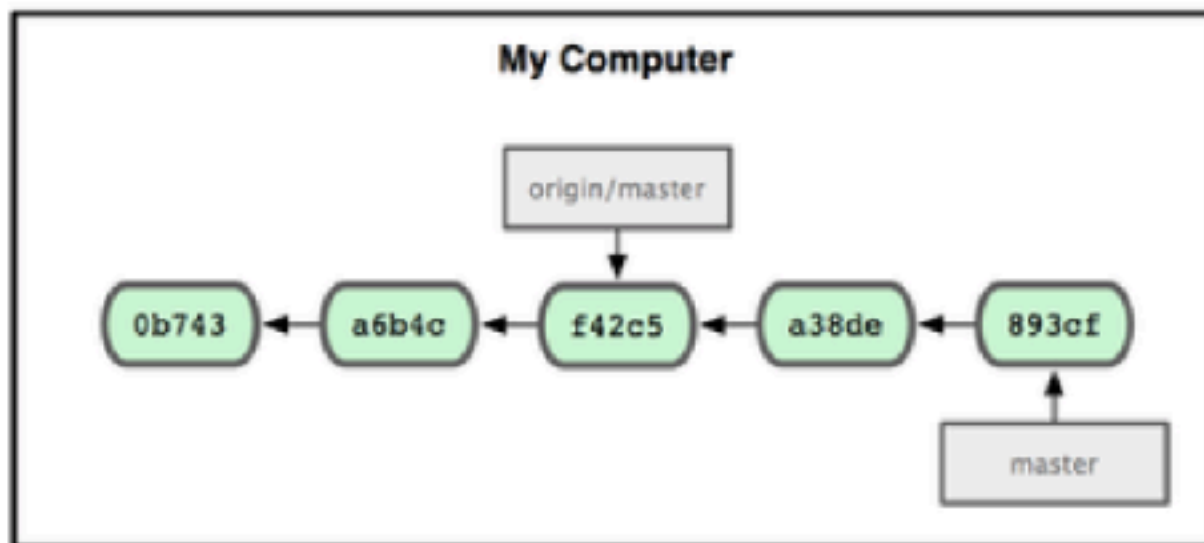
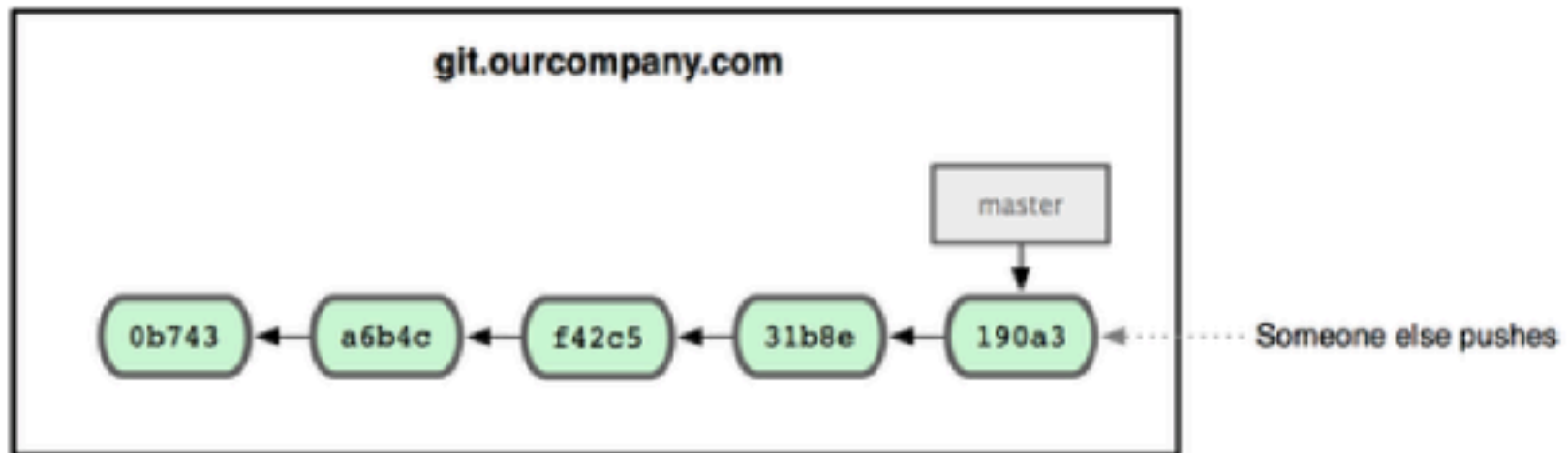
unmerged状态的文件

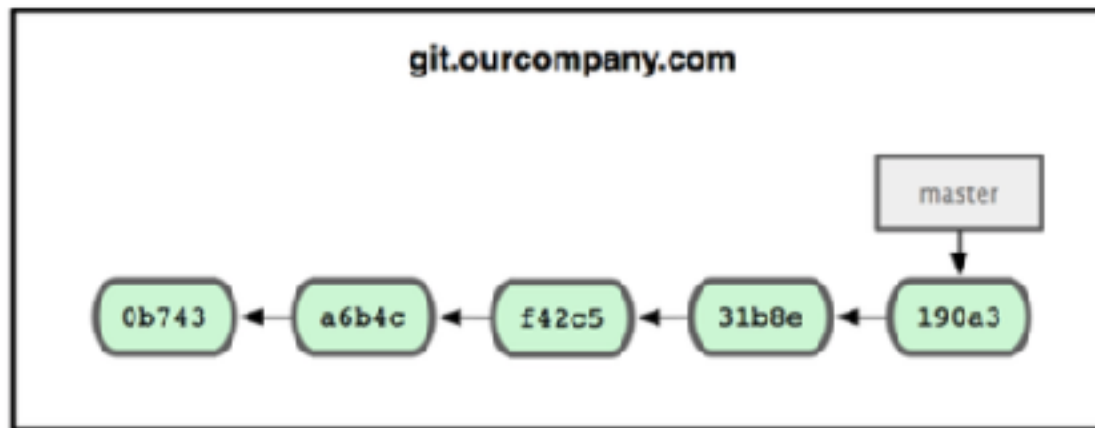
远程分支



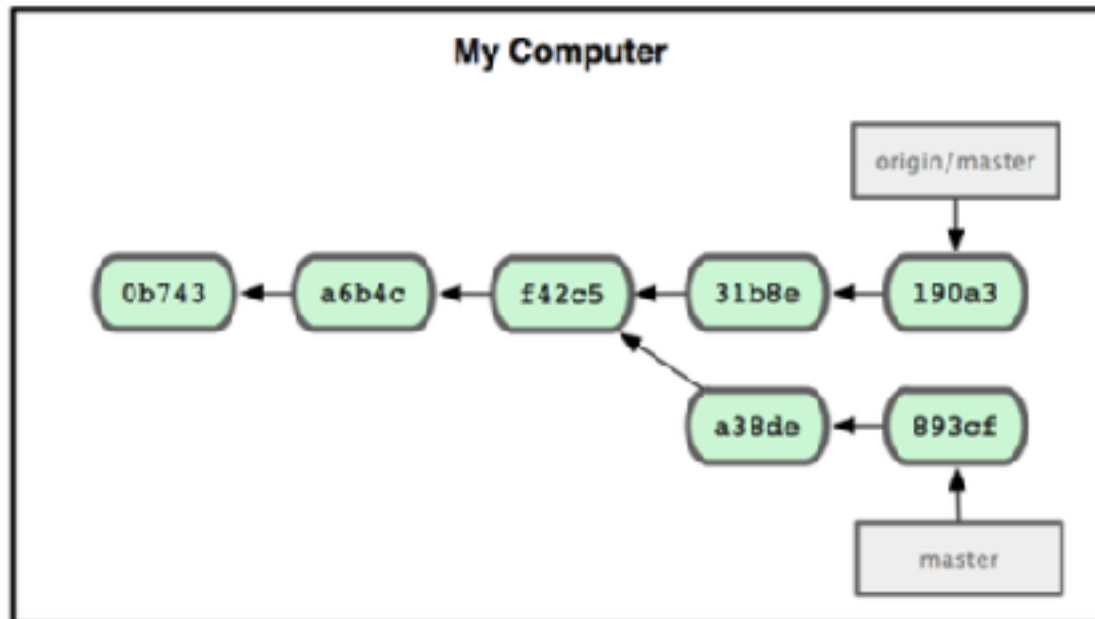
`git clone schacon@git.ourcompany.com:project.git`





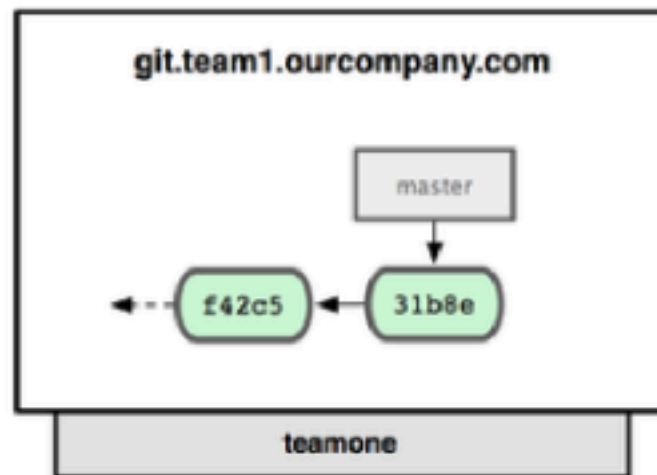
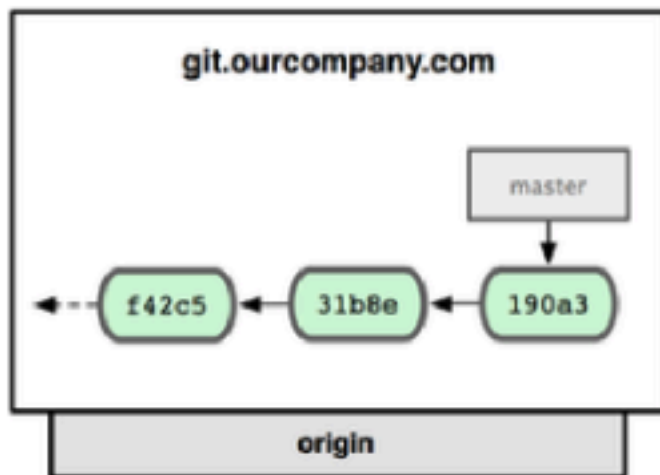


git fetch origin

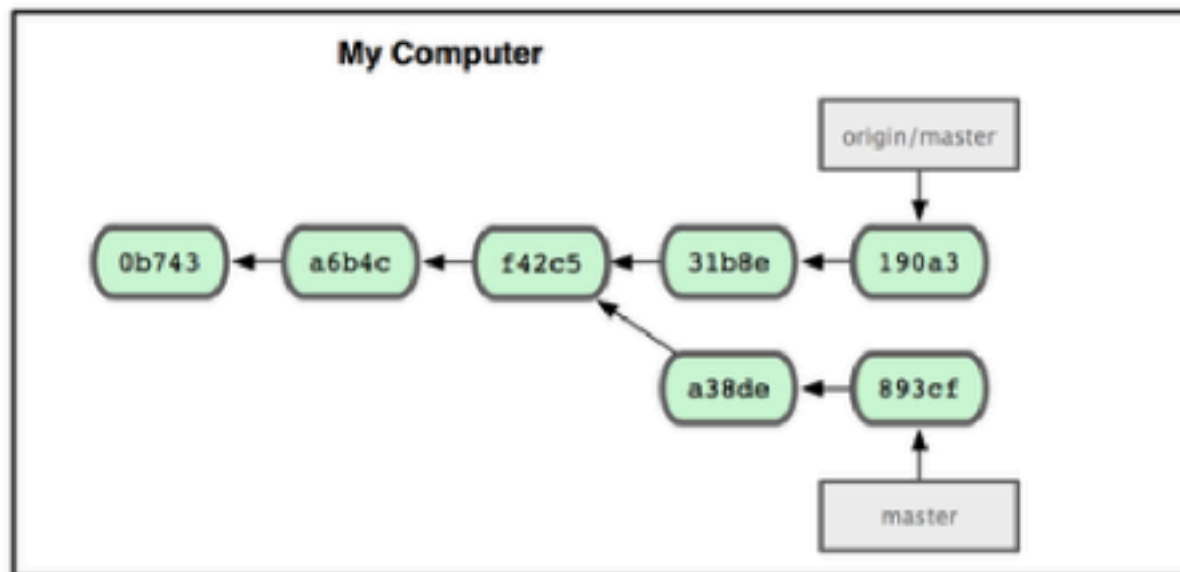


git fetch 命令会更新remote索引

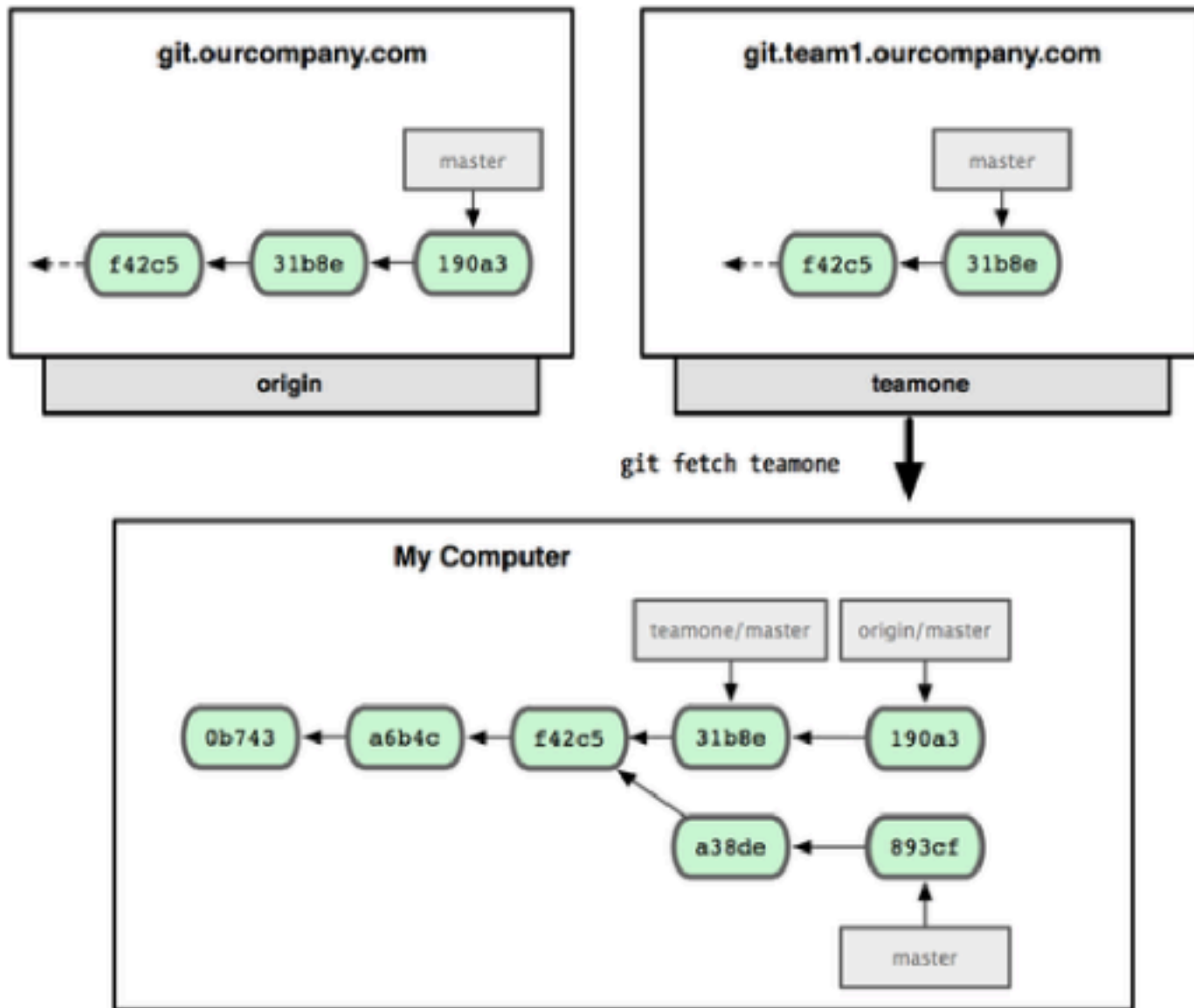
推送



```
git remote add teamone git://git.team1.ourcompany.com
```



把另一个服务器加为远程仓库



你在本地有了一个指向teamone服务器上master分支的索引

git push origin serverfix

跟踪分支

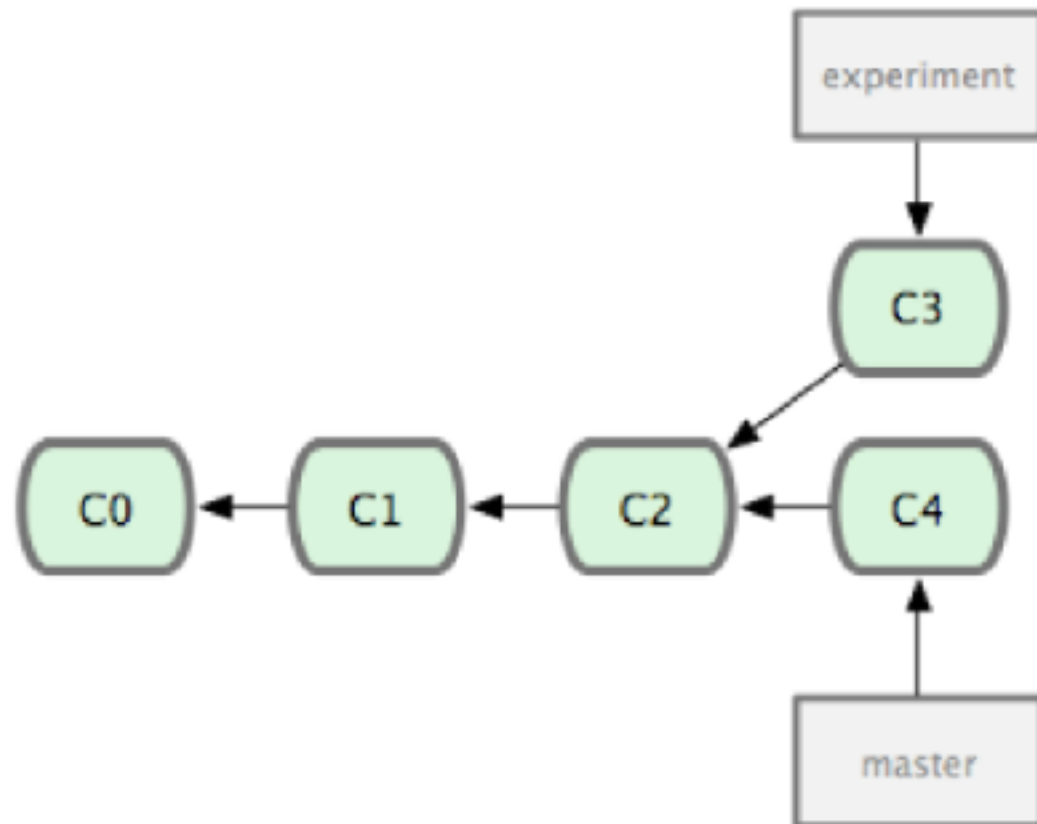
`git pull`

`git checkout -- track origin/serverfix`

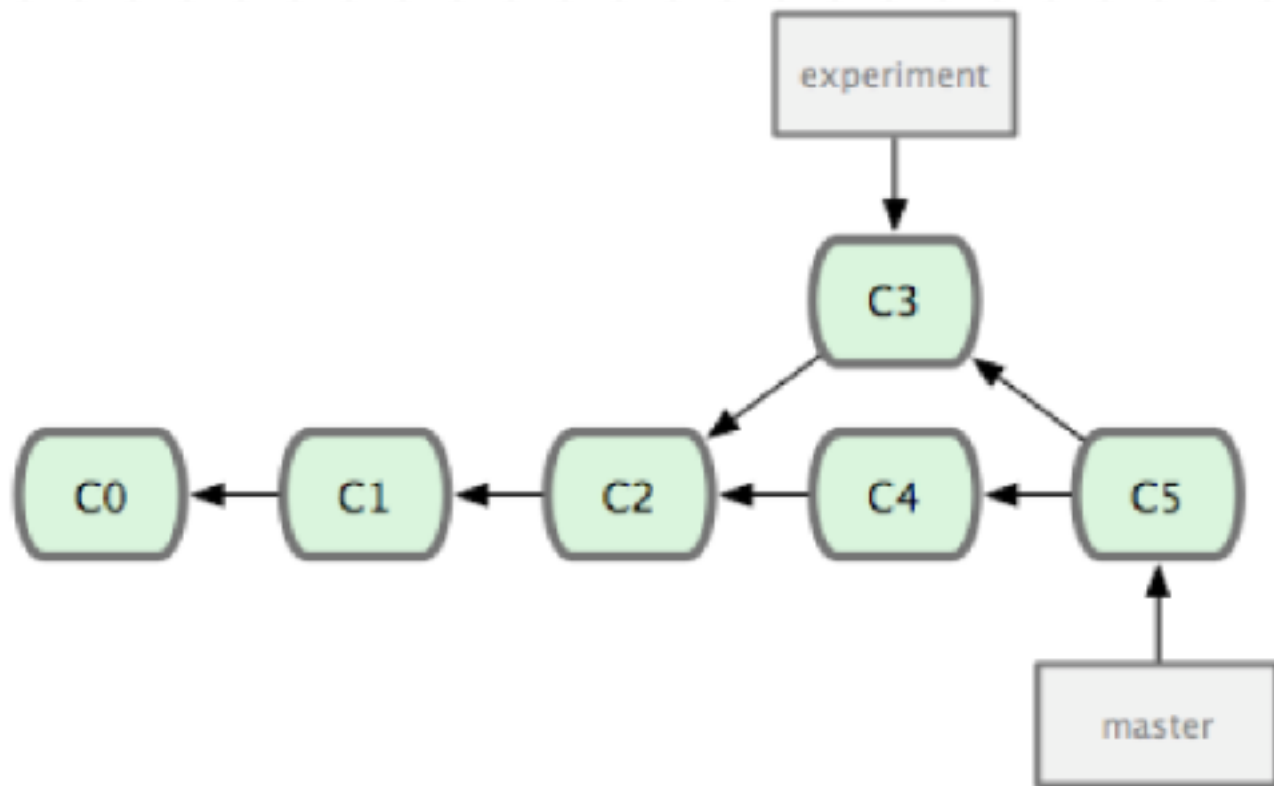
删除远程分支

```
git push origin :serverfix
```

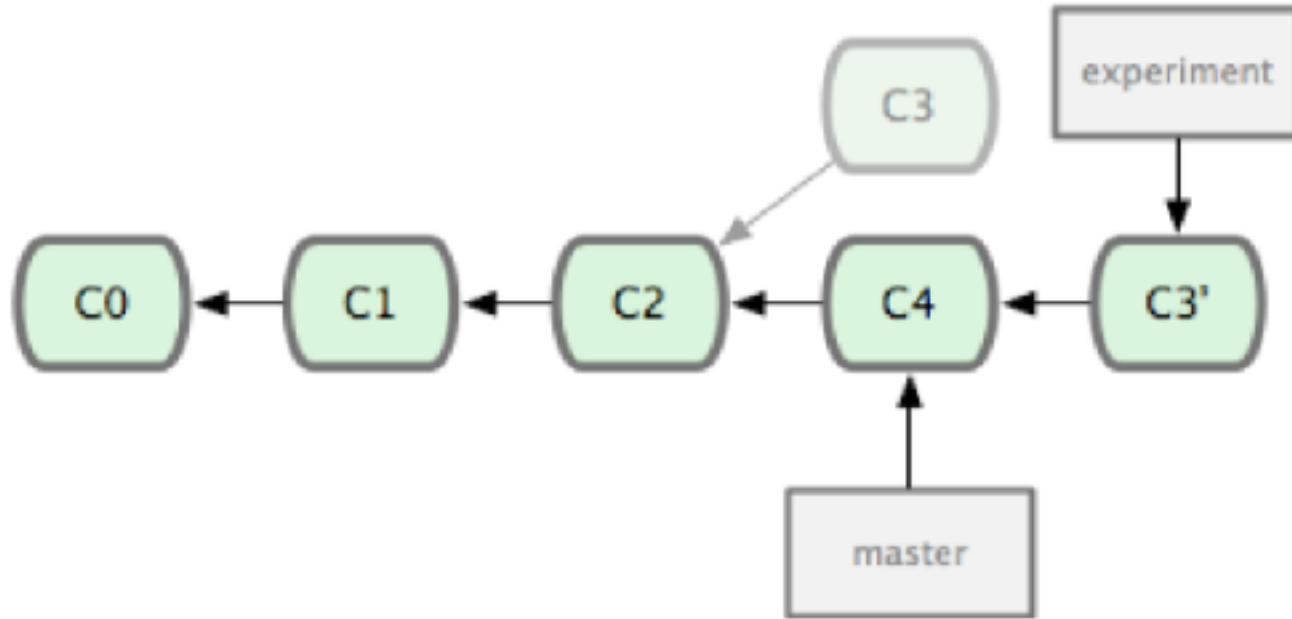
rebase (衍合)



最初分叉的提交历史



使用普通 merge 整合分叉历史



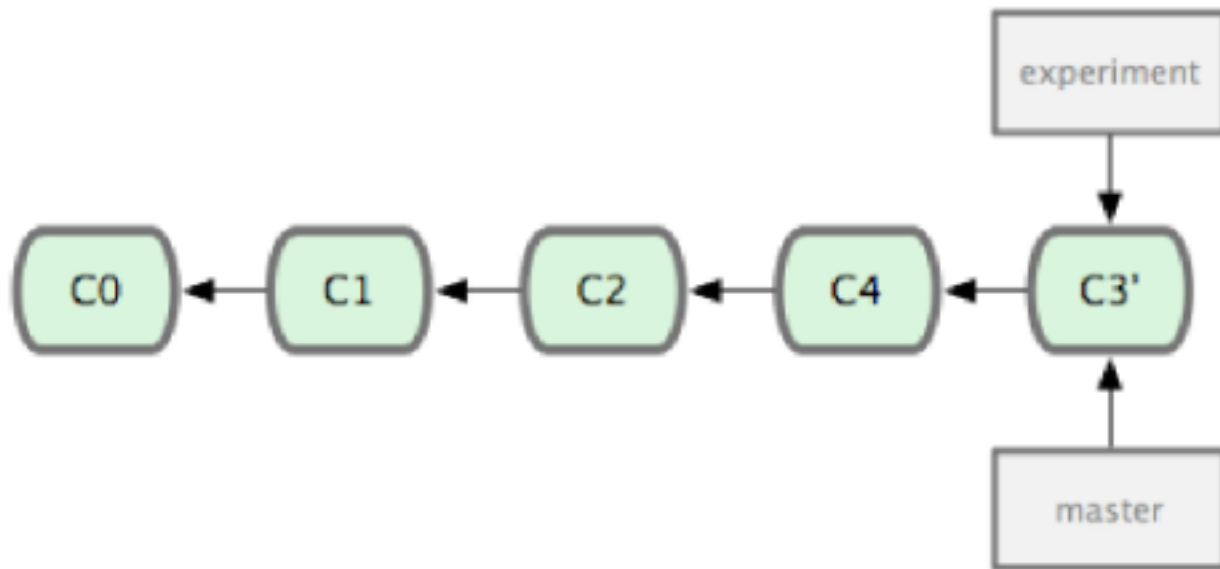
使用rebase, 把C3里产生的改变衍合到C4中

```
$ git checkout experiment
```

```
$ git rebase master
```

```
First, rewinding head to replay your work on top of it...
```

```
Applying: added staged command
```



回到master分支进行一次快进合并(merge)

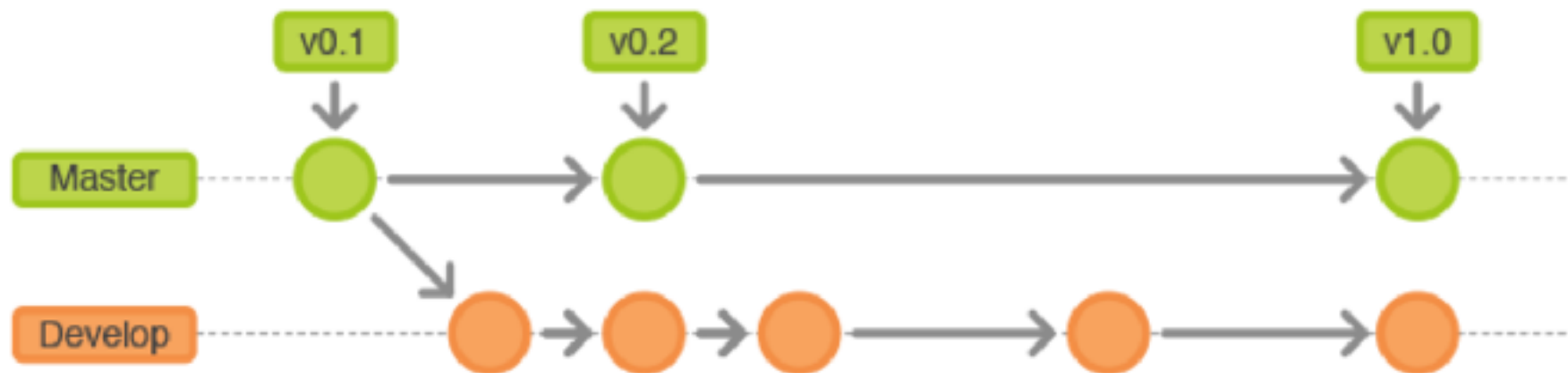
rebase的其他用法

- `git rebase -i`
- 像上帝一样控制历史

rebase(衍合)的风险

○永远不要衍合那些已经推送到公共仓库的更新

Gitflow 工作流



➤ 更多说明参见 Git工作流指南

<http://nvie.com/posts/a-successful-git-branching-model/>

<http://blog.jobbole.com/76867/>



maven-gitflow-plugin

- gitflow:feature-finish
- gitflow:feature-start
- gitflow:hotfix-finish
- gitflow:hotfix-start
- gitflow:release
- gitflow:release-finish
- gitflow:release-start

7. GUI客户端

- SmartGit <http://www.syntevo.com/smartgit/>
- 查看历史树&提交详细
 - 可以同时查看多个指定的分支或Tag
- 查看提交修改
- 比较指定2个版本的差异（即Code Review！）
- 调整提交
-

GUI客户端

- SmartGIT <http://www.syntevo.com/smartgit/>
- SourceTree <https://www.sourcetreeapp.com/>

资源索引

○ Git资料汇总 <https://github.com/xirong/my-git> 丰富!

○ **zsh** + [oh-my-zsh](#)

○ <https://github.com/oldratlee/why-git>

○ [Git工作流指南](#)

○ Git书籍 <http://www.douban.com/doulist/1686793/>

Thank you
&
QA

