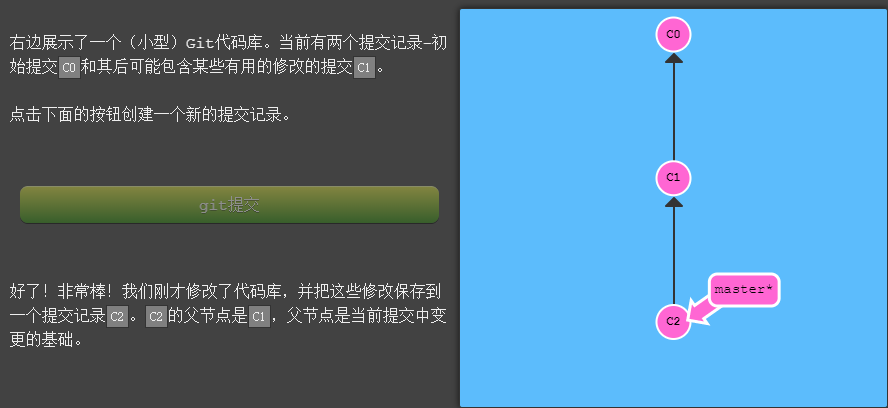
## 怎么使用git

问题：

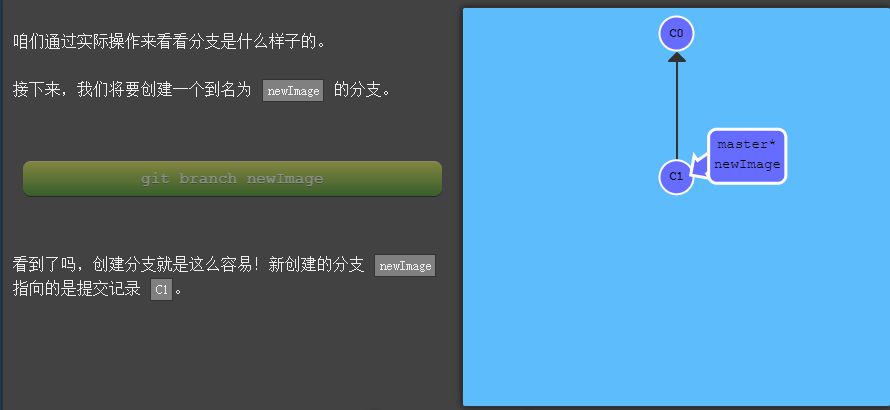
1. 如何下载git
2. 如何安装git
3. 如何新建git账号
4. 如何在远程服务器上搭建服务及git仓库
5. 如何新建本地git服务及本地仓库
6. 如何将本地仓库和远程仓库关联

### git commit（提交）

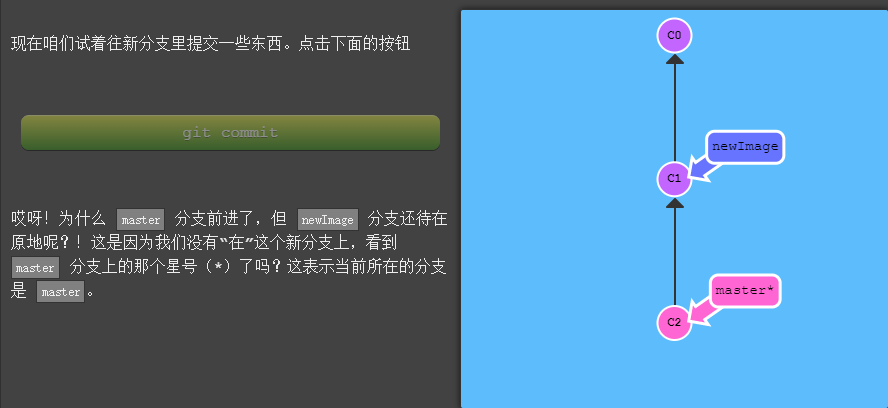
* + 1. Git仓库中的提交记录保存的是你的目录下所有文件的快照，就像是把整个目录复制，然后再粘贴一样，但比复制粘贴优雅很多！Git希望提交记录调整地轻量，因此在你每次进行进行提交时，它并不会盲目地复制整个目录。条件允许的情况下，它可以更改当前版本与仓库中的上一个版本进行对比，并把所有的差异打包到一起作为一个提交记录。Git还保存了提交的历史记录。这也是为什么大多数提交记录的上面都有父节点的原因-我们会在图示中用箭头来表示这种关系。对于项目组的成员来说，维护提交历史对大家都有好处。提交记录非常轻量，可以快速地在这些提交记录之间切换！



### **Git Branch(**新建一个分支**)**



git branch newImage 创建一个到名为 newImage 的分支



在master分支提交一次记录

### 切换分支



git checkout newImage切换到newImage 分支上

### 创建一个新分支同时切换到新创建的分支

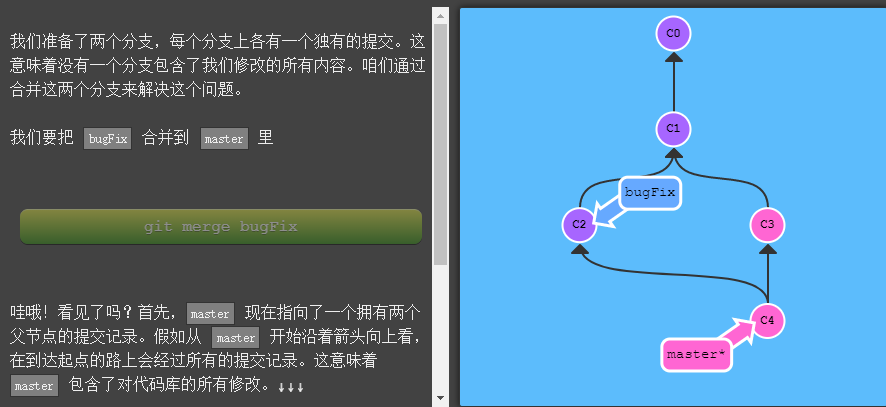
* 1. git checkout -b <your-branch-name>

### 分支与合并

#### 5.1第一种方法 —— git merge

在 Git 中合并两个分支时会产生一个特殊的提交记录，它有两个父节点

把bugFix分支合并到master中，相当于将目标目录复制到当前目录cp -r bugFix/\* ./



再把master分支合并到bugFix中

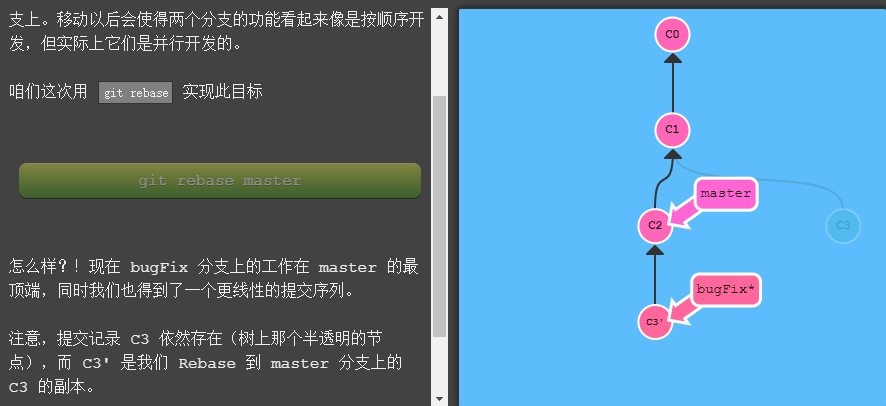


#### 5.2第**二种方法 —— git rebase**

Rebase 实际上就是取出一系列的提交记录，“复制”它们，然后在另外一个地方逐个的放下去

Git rebase master(等于是将当前目录复制到目标目录，相当于cp -r ./\* master)



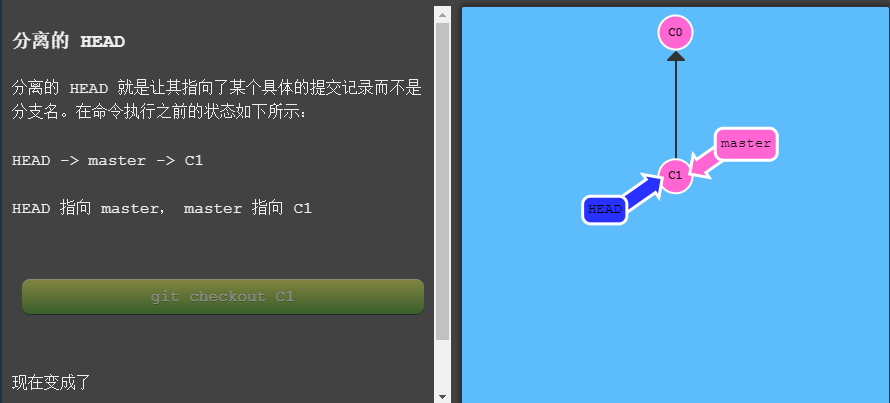


### HEAD分离 在提交树上移动

HEAD 是一个对当前检出记录的符号引用

—— 也就是指向你正在其基础上进行工作的提交记录。

HEAD 总是指向当前分支上最近一次提交记录



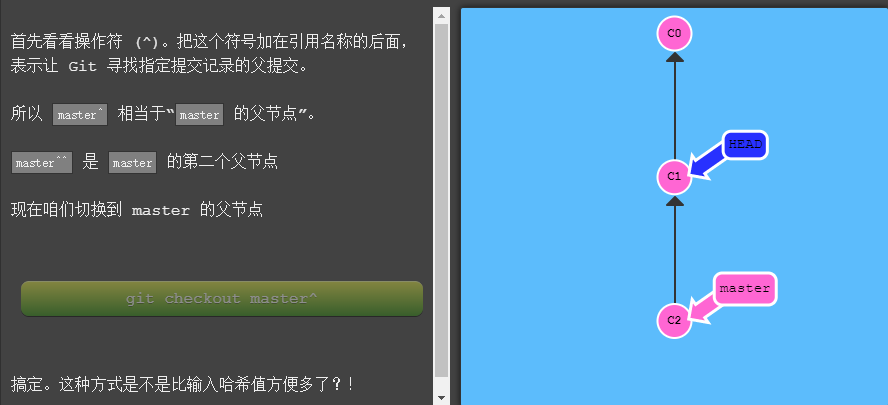
如果想看 HEAD 指向，可以通过**cat .git/HEAD** 查看，

如果 HEAD 指向的是一个引用，还可以用 **git symbolic-ref HEAD**查看它的指向

### 相对引用

在实际应用时，用**git log** 来查查看提交记录，通过哈希值指定提交记录很不方便，Git 引入了相对引用,使用相对引用的话，你就可以从一个易于记忆的地方（比如 bugFix 分支或 HEAD）开始计算。

#### 7.1使用 ^ 向上移动 1 个提交记录

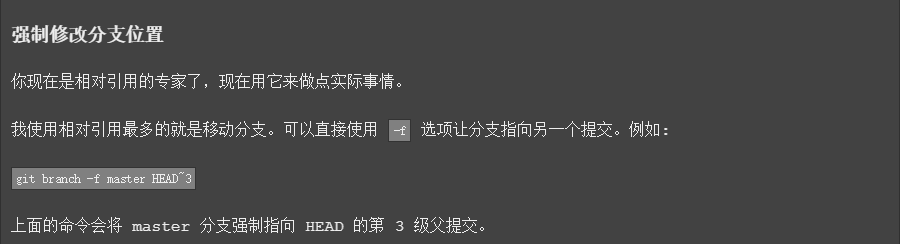


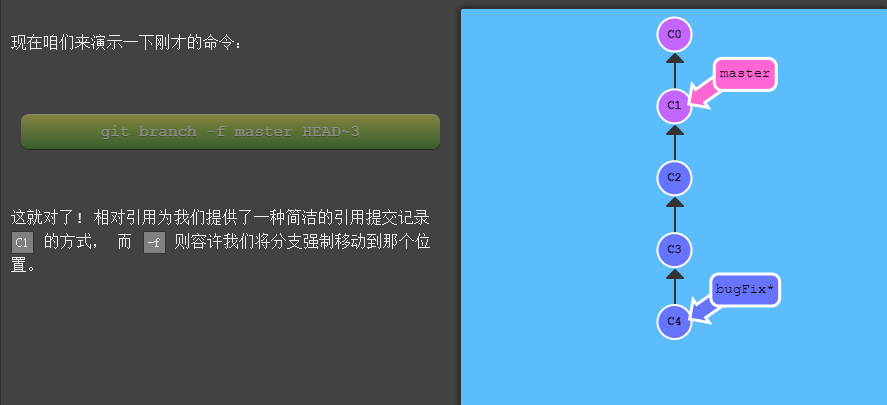
#### 7.2使用 ~<num> 向上移动多个提交记录，如 ~3



#### 7.3强制修改分支位置(git branch -f master HEAD~3)

\*\*\*\*还是不太理解





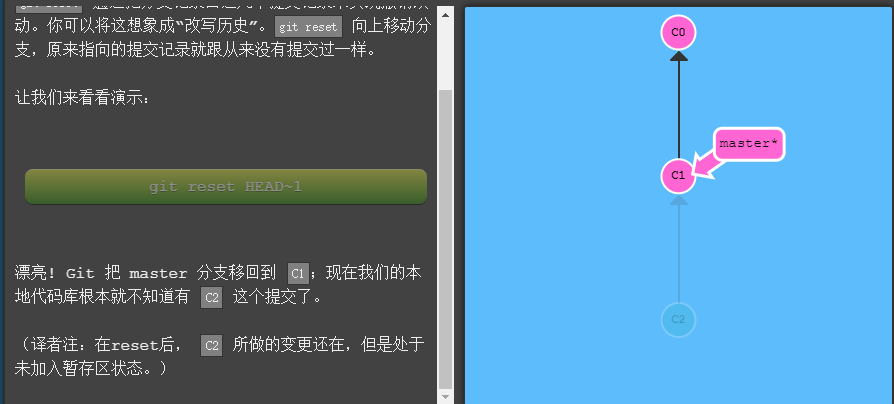
### 撤销变更

#### 8.1 git reset

git reset 通过把分支记录回退几个提交记录来实现撤销改动。

你可以将这想象成“改写历史”。

git reset 向上移动分支，原来指向的提交记录就跟从来没有提交过一样。



漂亮! Git 把 master 分支移回到 C1；现在我们的本地代码库根本就不知道有 C2 这个提交了。

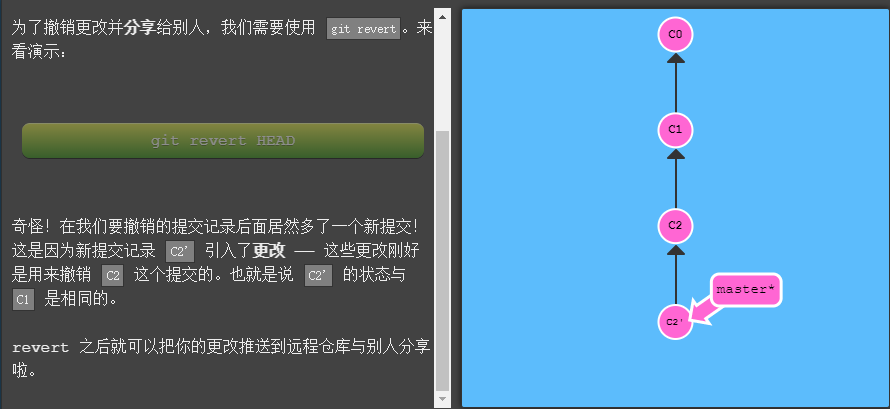
（译者注：在reset后， C2 所做的变更还在，但是处于未加入暂存区状态。）

问题：如何再切回C2呢？

#### 8.2 git revert

虽然在你的本地分支中使用 git reset 很方便，但是这种“改写历史”的方法对大家一起使用的远程分支是无效的哦！

为了撤销更改并分享给别人，我们需要使用 git revert



问题：如果远程的C2未来的及撤销便有同事提交了新的记录，我再revert C2 那么图应该是怎么样的？