## 怎么使用git

问题：

1. 如何下载git

二、配置步骤

1、安装git 服务端

Linux作为服务器端系统，Windows作为客户端系统，分别安装Git

服务器端

[admin@ceontOS ~]$ su root                  #切换到root用户名

Password:                                   #输入root用户的密码

[root@ceontOS ~]# yum install -y git        #执行该命令进行Git安装

安装完后，查看Git版本

[root@ceontOS ~]# git --version

1. 如何安装git

客户端：

下载 Git for Windows，地址：https://git-for-windows.github.io/

安装完之后，可以使用 Git Bash 作为命令行客户端。

安装完之后，查看 Git 版本

$ git --version

git version 2.11.0.windows.1

https://img-blog.csdnimg.cn/20190409153613289.png

1. 如何新建git账号

服务器端创建 git 用户，用来管理 Git 服务，并为 git 用户设置密码

[root@ceontOS ~]# cd /home                                  #进入/home/目录

[root@ceontOS home]# id git                                 #查看git用户是否存在

id: git: no such user                                       #提示git用户不存在

[root@ceontOS home]# useradd git                            #创建git用户

[root@ceontOS home]# passwd git                             #为git用户创建密码

Changing password for user git.

New password:                                               #设置密码

BAD PASSWORD: The password is shorter than 8 characters

Retype new password:                                        #确认密码

passwd: all authentication tokens updated successfully.

1. 如何修改git用户名和邮箱以及密码

查看当前用户名和邮箱

git config user.name

git config user.email

修改

git config --global user.name "zhangsan(新的用户名)"

git config --global user.email "123456@qq.com(新的邮箱)"

git config --global user.password "123456(新的密码)"

//这里的zhangsan和邮箱都是你修改之后的用户名和邮箱

1. 如何在远程服务器上搭建服务及git仓库

 git init --bare ./fenlin88l.git    #这步很重要，初始化项目测试目录

chown -R git:git fenlin88l.git/    #将拥有者改为git用户

1. 如何新建本地git服务及本地仓库

git clone  git@服务器公网IP地址:/home/git/fenlin88l.git    #IP地址后面跟冒号，冒号后面是刚才初始化的项目文件夹的绝对路径

![](http://upload-images.jianshu.io/upload\_images/2267589-7bedba534bdc9f04.png?imageMogr2/auto-orient/strip%7CimageView2/2/w/1240)

例:  git clone git@47.107.193.147:/home/git/fenlin88l.git

当第一次连接到目标 Git 服务器时会得到一个提示：

The authenticity of host '118.178.142.77 (118.178.142.77)' can't be established.

ECDSA key fingerprint is SHA256:JwC9NxLIjBGqtLC2NUk8MulSc3XH3mM5AWMcFz0a5/8.

Are you sure you want to continue connecting (yes/no)? yes

选择 yes：

Warning: Permanently added '118.178.142.77' (ECDSA) to the list of known hosts.

此时 C:\Users\用户名\.ssh 下会多出一个文件 known\_hosts，以后在这台电脑上再次连接目标 Git 服务器时不会再提示上面的语句。

【说明】如果你的服务器没有配置SSH连接，那么按照正常情况会让你输入git用户的密码，输入正确后就能进行项目克隆了。

![](http://upload-images.jianshu.io/upload\_images/2267589-776f44a088bba653.png?imageMogr2/auto-orient/strip%7CimageView2/2/w/1240)

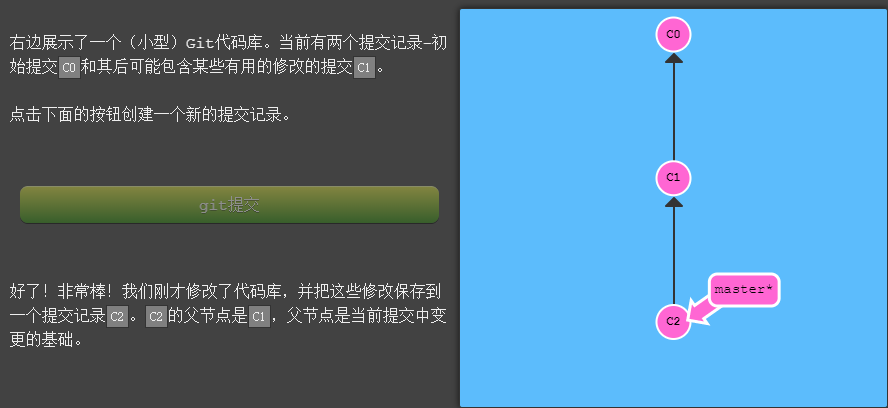
如果不采用 SSH 公钥来进行验证，则每次都要输入密码，很麻烦，下面就来配置SSH公钥验证的方式来clone项目

   问题：如果不配置ssh公钥使用git远程仓库会怎么样，会出现以下：

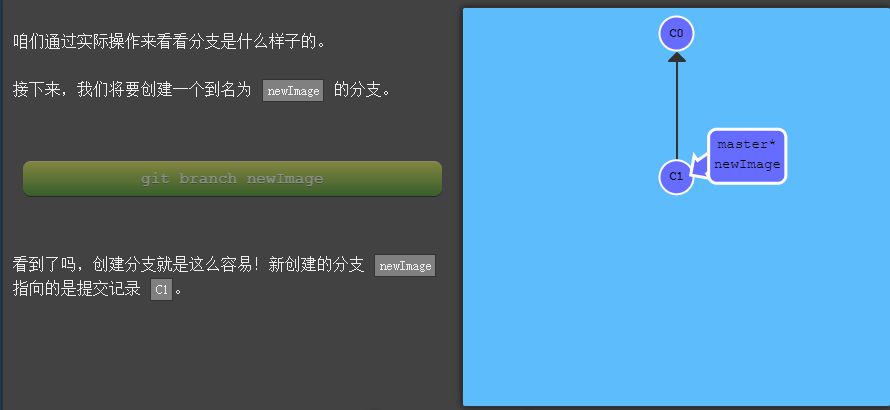
1. 如何将本地仓库和远程仓库关联

### git commit（提交）

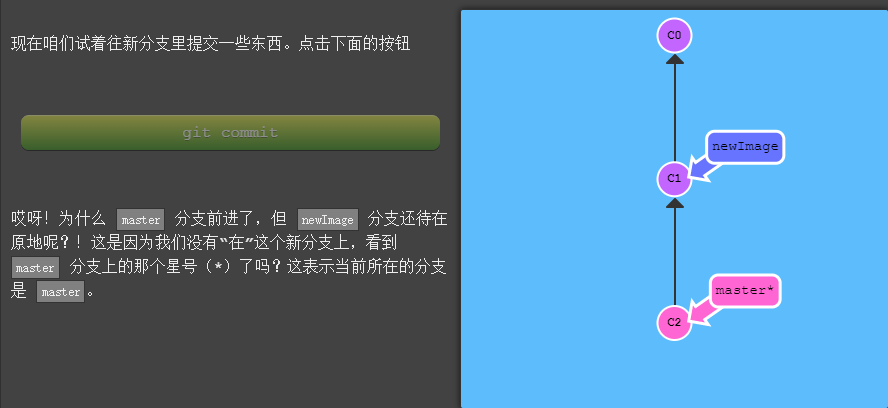
* + 1. Git仓库中的提交记录保存的是你的目录下所有文件的快照，就像是把整个目录复制，然后再粘贴一样，但比复制粘贴优雅很多！Git希望提交记录调整地轻量，因此在你每次进行进行提交时，它并不会盲目地复制整个目录。条件允许的情况下，它可以更改当前版本与仓库中的上一个版本进行对比，并把所有的差异打包到一起作为一个提交记录。Git还保存了提交的历史记录。这也是为什么大多数提交记录的上面都有父节点的原因-我们会在图示中用箭头来表示这种关系。对于项目组的成员来说，维护提交历史对大家都有好处。提交记录非常轻量，可以快速地在这些提交记录之间切换！



### 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

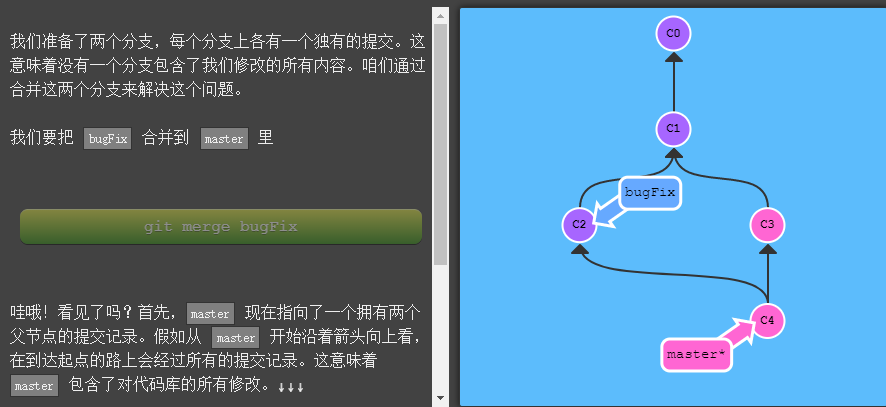
* 1. git checkout -b <your-branch-name>

### 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

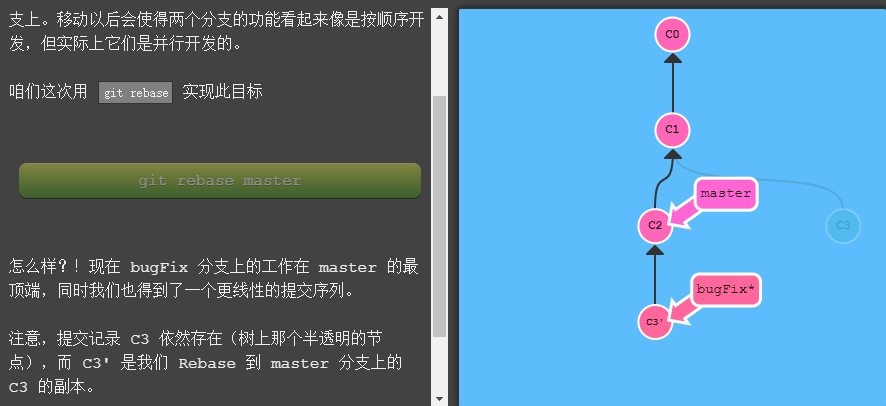


#### 5.2第二种方法 —— git rebase

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)



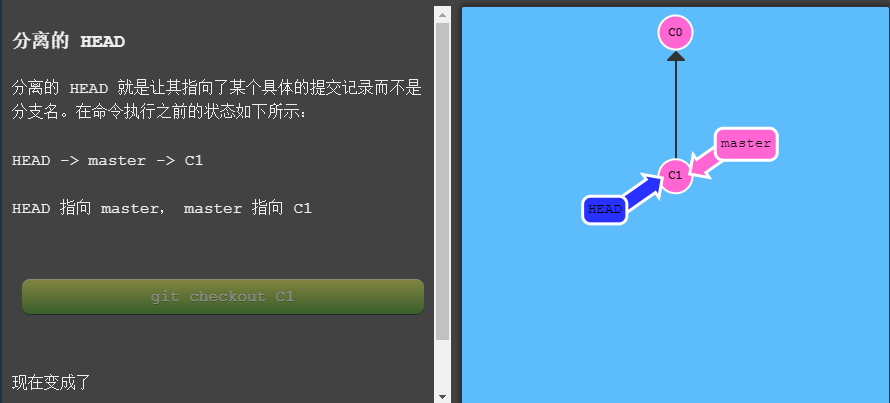


### HEAD分离 在提交树上移动

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



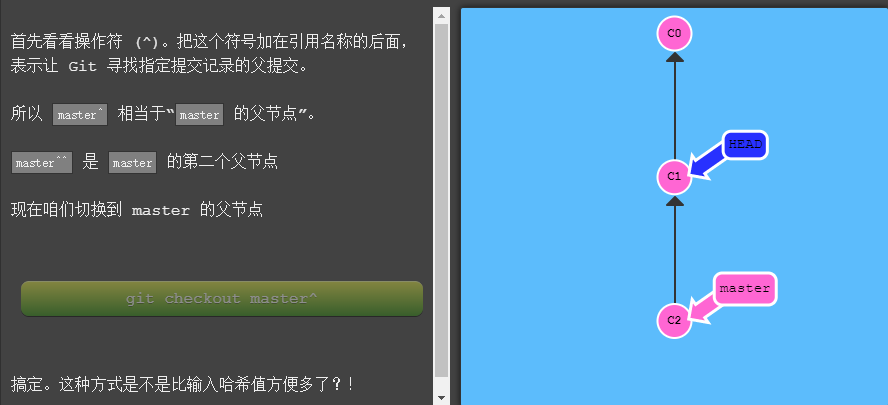
如果想看 HEAD 指向，可以通过**cat .git/HEAD** 查看，

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

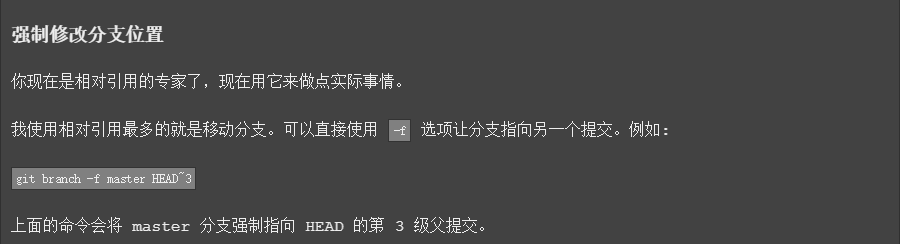


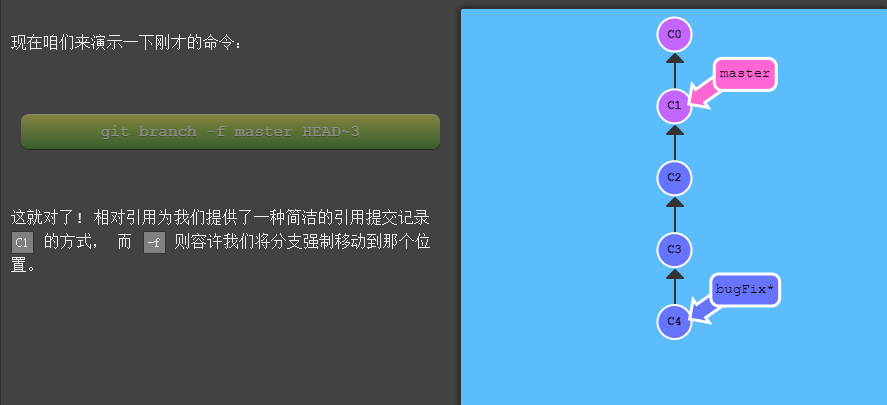
#### 7.2使用 ~<num> 向上移动多个提交记录，如 ~3



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

\*\*\*\*还是不太理解





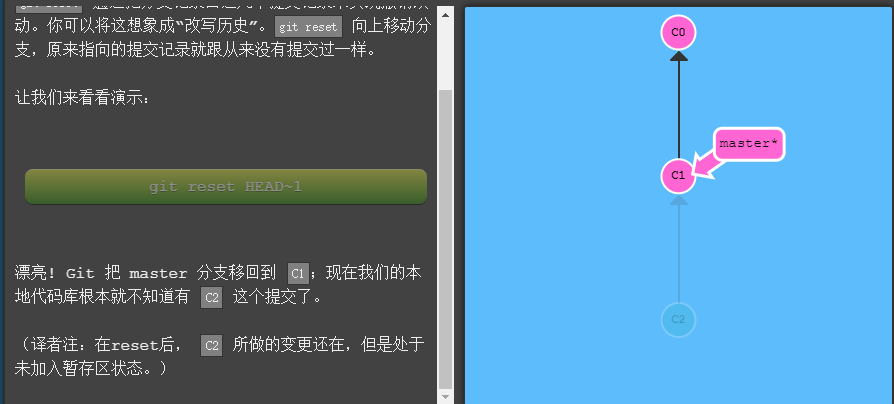
### 撤销变更

#### 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。

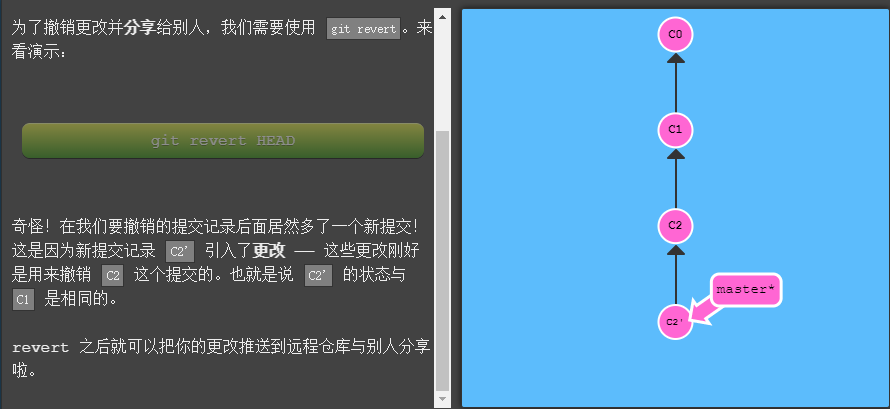
（译者注：在reset后， C2 所做的变更还在，但是处于未加入暂存区状态。）

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？

### 9.自由修改提交树

整理提交记录

1: git cherry-pick, 命令形式为:

git cherry-pick <提交号>...



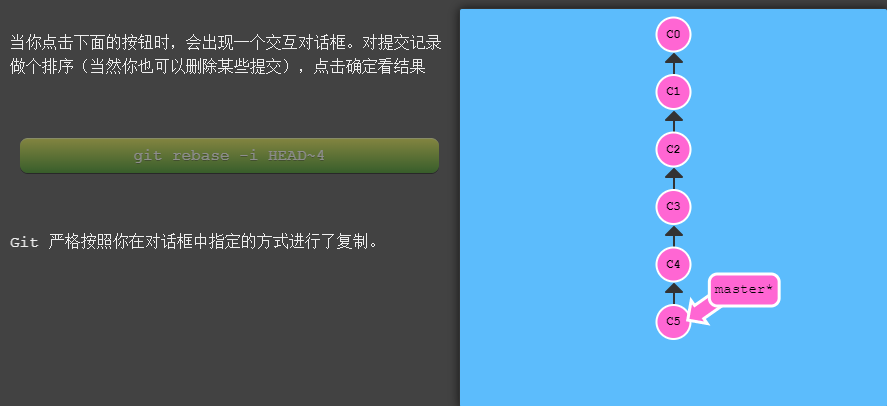
### 10.交互式的 rebase

交互式 rebase 指的是使用带参数 --interactive 的 rebase 命令, 简写为 –i

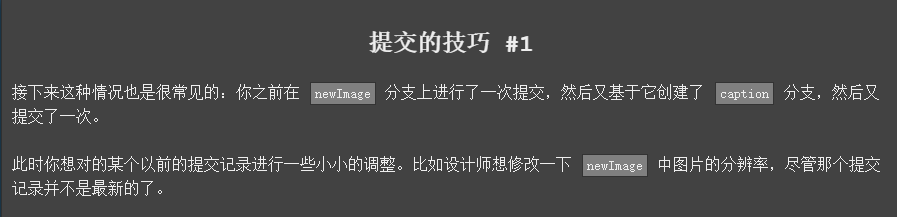
调整提交记录的顺序（通过鼠标拖放来完成）

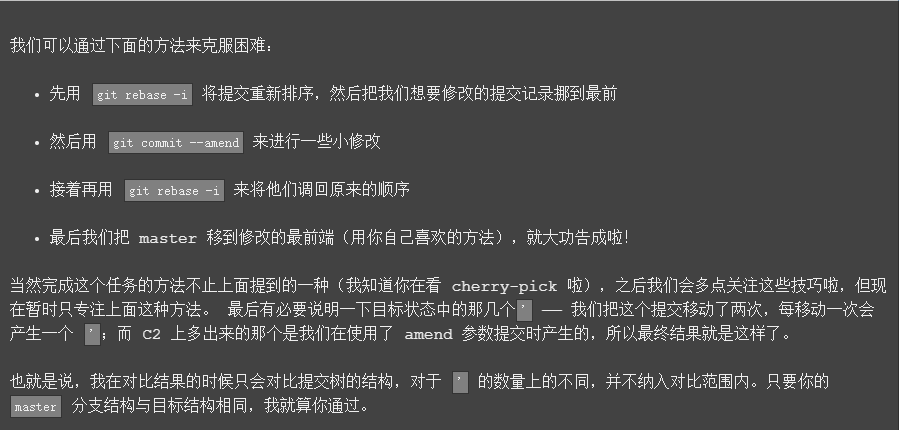
删除你不想要的提交（通过切换 pick 的状态来完成，关闭就意味着你不想要这个提交记录）

合并提交。 遗憾的是由于某种逻辑的原因，我们的课程不支持此功能，因此我不会详细介绍这个操作。简而言之，它允许你把多个提交记录合并成一个。



#### 10.1 提交技巧1 --amend



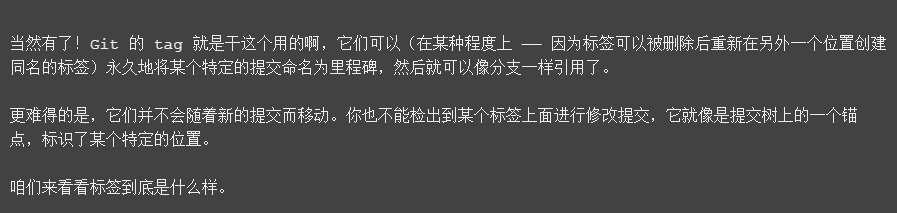


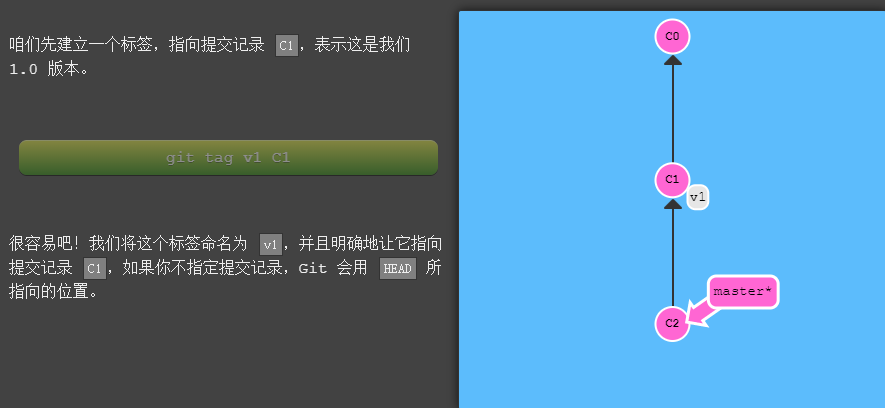
#### 10.2 提交技巧2 cherry-pick



### 11. Git Tags







### 12. Git Describe

Git 还为此专门设计了一个命令用来描述离你最近的锚点（也就是标签），它就是 git describe！

Git Describe 能帮你在提交历史中移动了多次以后找到方向；当你用 git bisect（一个查找产生 Bug 的提交记录的指令）找到某个提交记录时，或者是当你坐在你那刚刚度假回来的同事的电脑前时， 可能会用到这个命令。

git describe 的​​语法是：

git describe <ref>

<ref> 可以是任何能被 Git 识别成提交记录的引用，如果你没有指定的话，Git 会以你目前所检出的位置（HEAD）。

它输出的结果是这样的：

<tag>\_<numCommits>\_g<hash>

tag 表示的是离 ref 最近的标签， numCommits 是表示这个 ref 与 tag 相差有多少个提交记录， hash 表示的是你所给定的 ref 所表示的提交记录哈希值的前几位。

当 ref 提交记录上有某个标签时，则只输出标签名称

