**分布式版本控制系统Git**

**课程讲义**

**主讲：郭磊**

**2018**

**专题：分布式版本控制系统Git**

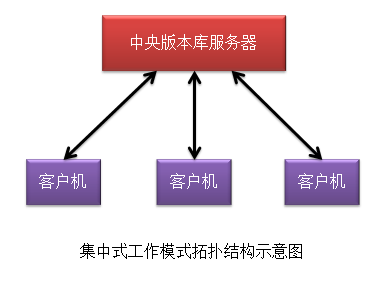
# Git概述

## 版本控制系统概述

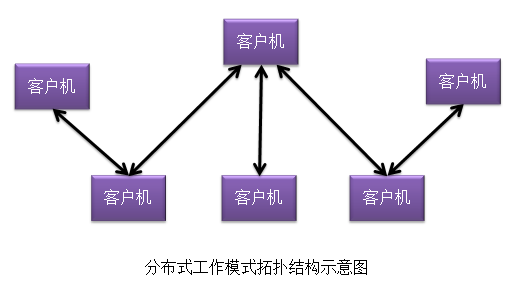
### 工作模式

版本控制系统的工作模式有两种：集中式工作模式与分布式工作模式。

#### 集中式工作模式



#### 分布式工作模式



### 常见的版本控制系统

CVS

SVN

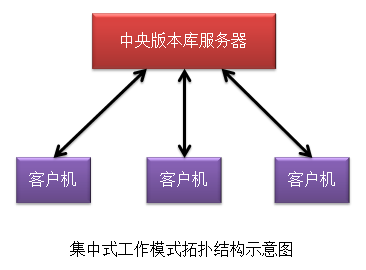
Git

## Git简介

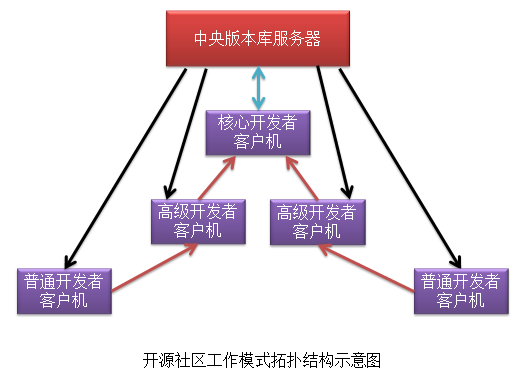
## Git的工作模式

针对不同性质的项目，Git可以采用不同的工作模式。

### 集中式工作模式

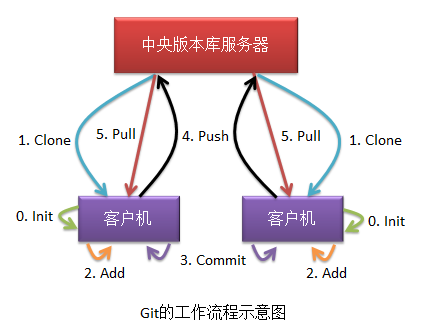


### 开源社区工作模式

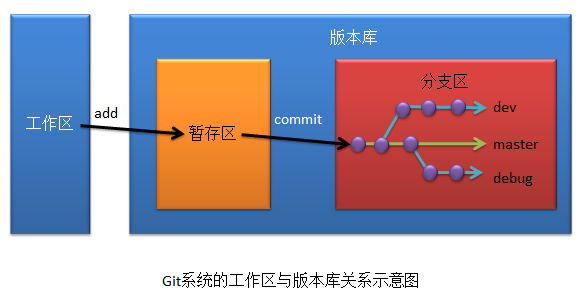


## Git的工作原理

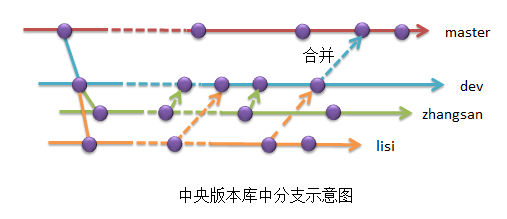
### Git的工作流程



### 工作区与版本库



### 分支的作用



## Git的下载与安装

### Git的下载

打开Git官网： <http://git-scm.com>

### Git的安装

### Git Bash简介

## 注意事项

### 版本控制的文件类型

对于版本控制系统，无论是CVS、SVN还是Git，都只能控制文本文件，对于二进制文件是无法控制的。不幸的是，微软的Office文件都属于二进制文件，不能使用版本控制系统进行版本控制。

### 字符编码与记事本

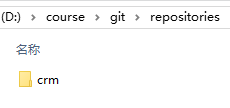
对于使用版本控制系统管理的文本文件，建议使用同一种字符编码UTF-8。

# Git本地版本库

## Git的初始化

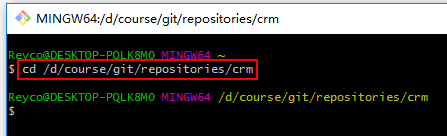
### 创建本地版本库

本例首先在D:\course\下新建一个git目录，然后在其中再新建一个repositories目录，其下将来可能会存在很多项目目录。每一个项目目录就是一个本地版本库。例如，有一个crm项目目录。

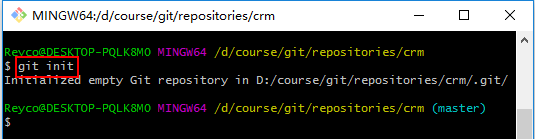


### 初始化版本库

#### 进入本地版本库目录

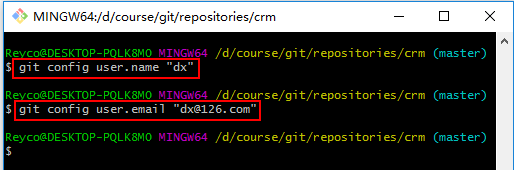


#### 执行初始化命令git init

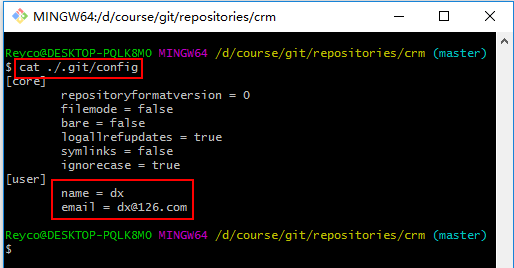


### 创建用户

只能在当前的本地库目录下运行该创建命令。本例创建本地库用户dx。

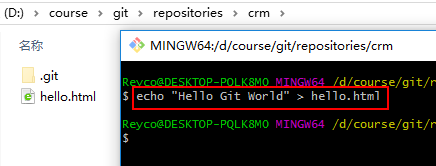


用户注册信息会写到当前版本库目录下的.git目录的config文件中。

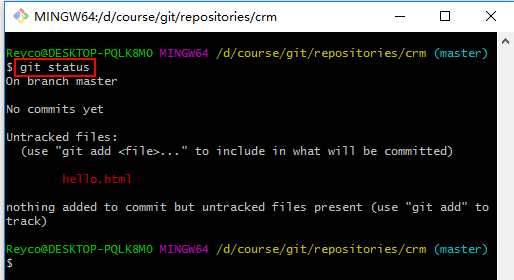


## 基本操作

### 新建文件



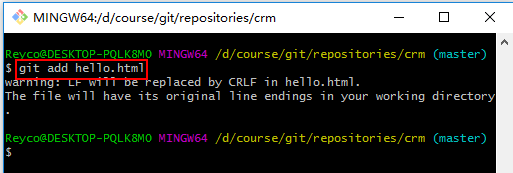
### 查看状态git status



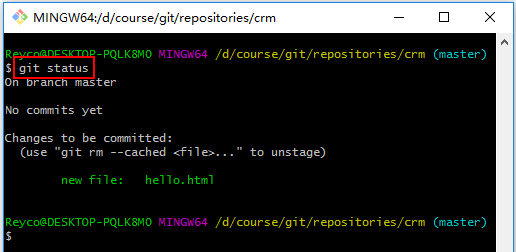
### 添加管理git add

该命令的作用是告诉Git系统，将指定文件的当前快照写入到版本库暂存区。即，将文件交给Git进行版本管理。

#### 添加一个文件



此时再次查看Git状态，提示，尚未提交。

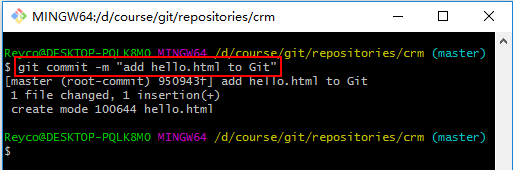


#### 添加多个文件

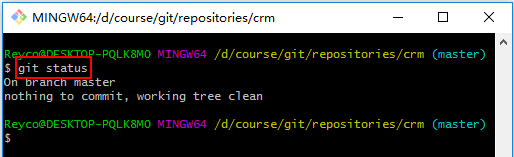
git add命令可以以多种方式一次性添加多个文件：

* add后的多个文件之间用空格分隔
* add后使用通配符\*指定多个文件
* add后指定目录，会将该目录中所有文件全部添加。将目录改为点(.)，表示将当前目录中所有文件全部添加

### 提交操作git commit



再次查看Git状态：

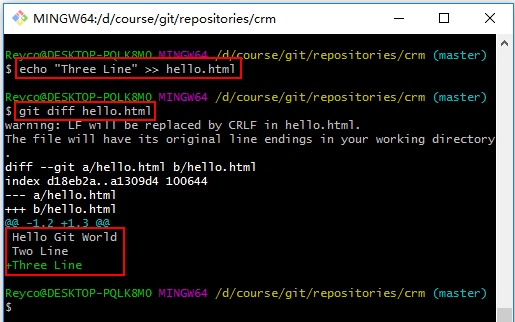


### 忽略文件

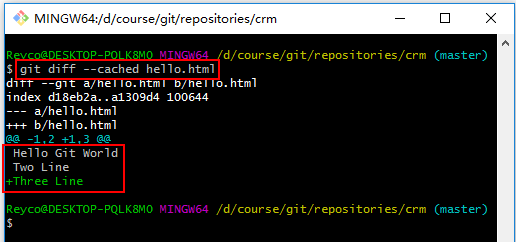
只需在工作区创建一个文件，名称为.gitignore，把要被忽略的文件名写入其中，然后将.gitignore文件add并commit到本地版本库即可。

### 查看区别

#### 比较工作区与暂存区git diff



#### 比较暂存区与分支区git diff --cached



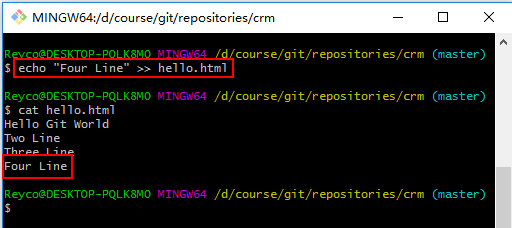
### 撤销修改

对于已修改过的文件内容需要撤销，根据修改内容已经出现的位置可以分为三种情况：

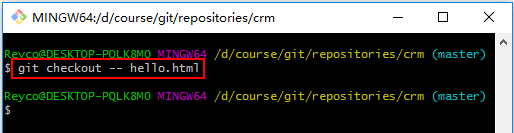
* 仅仅是工作区中内容进行了修改，还未add到暂存区
* 已经add到暂存区，但还未commit到版本库
* 已经commit到了版本库

不同的情况，具有不同的撤销方式。

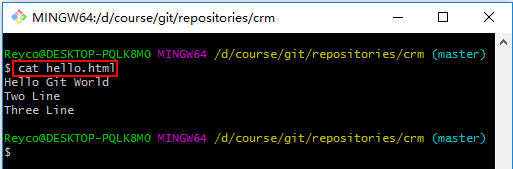
#### 仅在工作区中修改git checkout -- <file>



然后使用命令方式撤销之前修改。

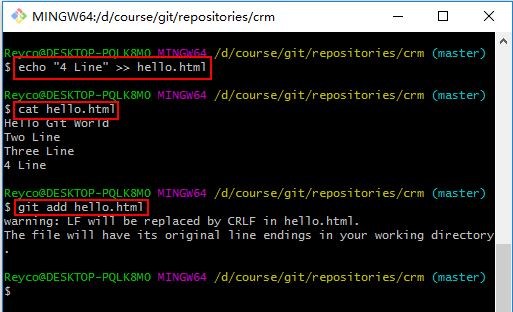


再查看文件内容，发现已经复原，撤销了之前的修改。

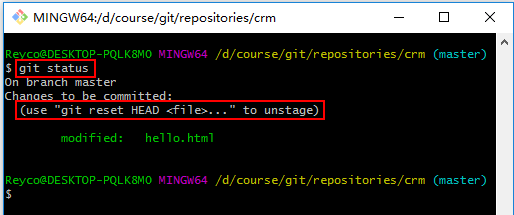


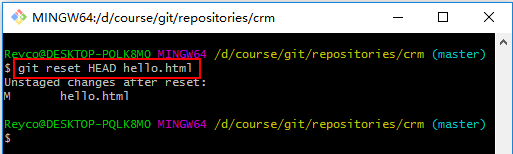
#### 已add到暂存区git reset HEAD

首先要修改文件内容，并将其add到暂存区。

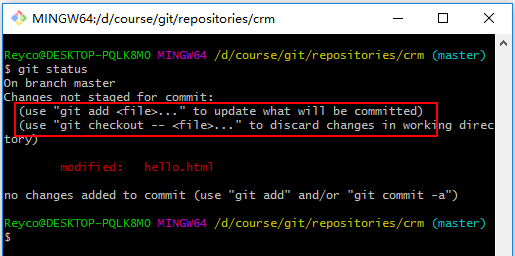


查看Git状态





再次查看Git状态，提示，可以使用git add命令将修改过的文件添加到暂存区，也可以使用git checkout -- 命令将工作区中的文件修改撤销。

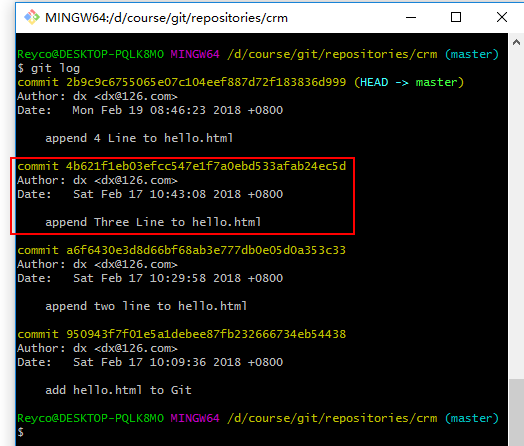


## 重置到之前版本

### 版本重置基础

#### 查看历史版本git log

##### 基本命令

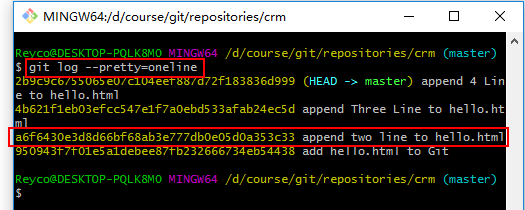


##### 翻页与退出

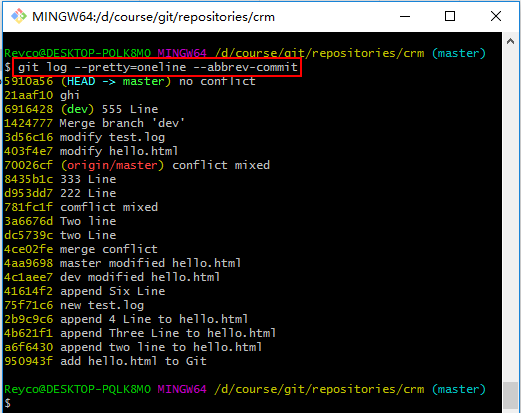
* 回车：显示下一行
* 空格：显示下一页
* q：退出git log命令

该命令对比Linux中的more命令可知，该命令的底层定是通过more命令读取的版本提交日志文件。

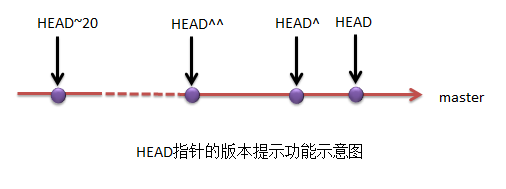
##### 单行显示--pretty=oneline



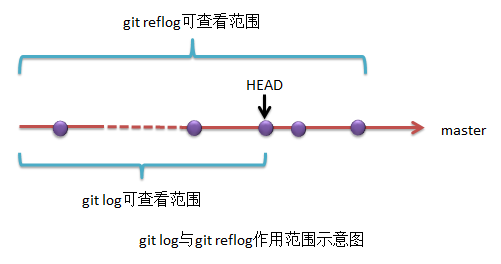
##### 单行简写显示--abbrev-commit

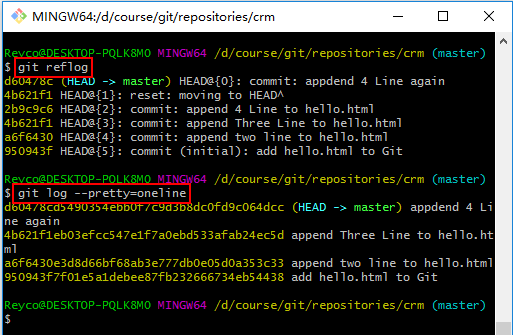


#### HEAD指针



#### 查看可引用历史版本git reflog

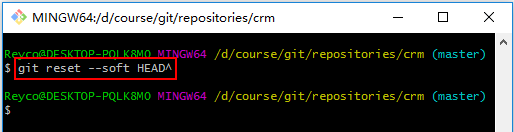




### 版本重置

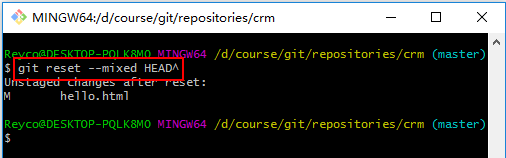
#### 软重置git reset --soft

仅仅重置了分支区的版本，工作区与暂存区中文件的版本并未发生重置。

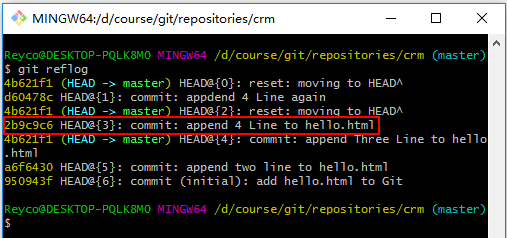


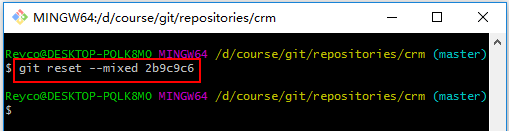
#### 混合重置git reset --mixed

该命令会首先使分支区重置到指定版本，然后再使暂存区内容与分支区内容一致。即分支区与暂存区均重置到了指定版本。不过，工作区未发生重置。



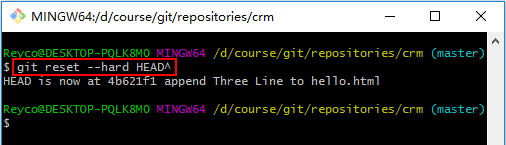
恢复重置前版本



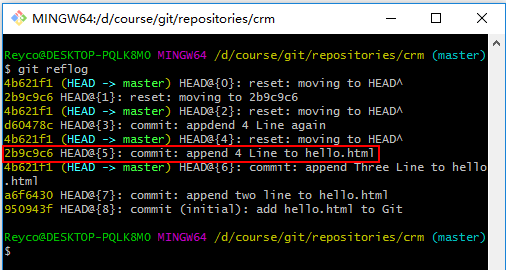


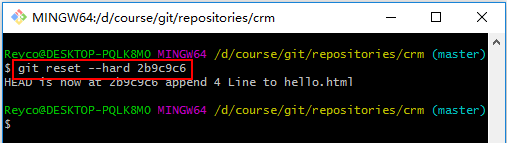
#### 硬重置git reset --hard

使分支区、暂存区与工作区内容都重置到指定版本。



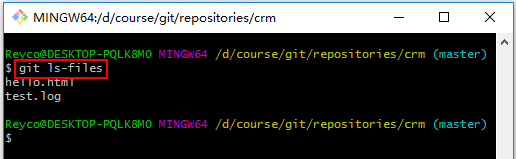
恢复重置前版本





## 删除文件

### 查看暂存区文件列表git ls-files

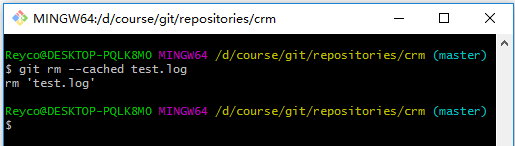


### 查看Git管理文件列表git ls-files --with-tree=HEAD

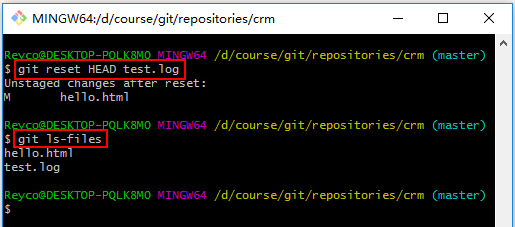


### 仅删除暂存区中指定文件git rm --cached <file>

该命令仅可删除暂存区中的指定文件。

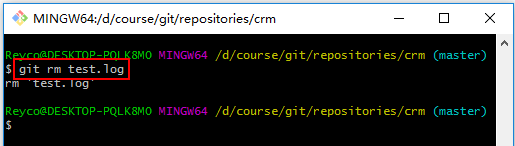


### 恢复被删文件git reset HEAD <file>



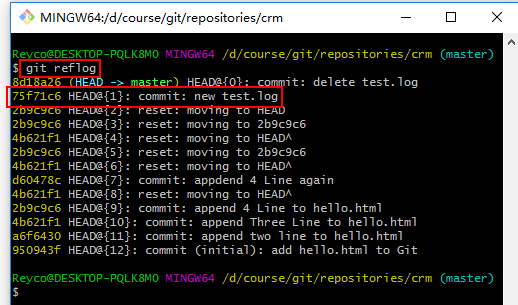
### 完全删除git rm

所谓完全删除，指的是将工作区与暂存区中指定文件删除，但并没有删除分支区该文件。若要将分支区该文件也删除，在完全删除后直接进行commit即可。

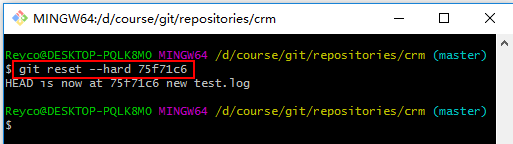


### 恢复被删文件

首先要查看要恢复的版本的commit-id。



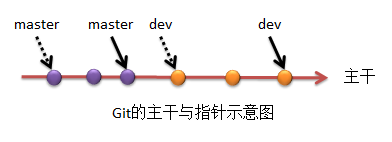
重置到该版本。由于要将三个区中的文件均做重置，所以这里需要使用硬重置。



## 分支管理

### Git分支理论

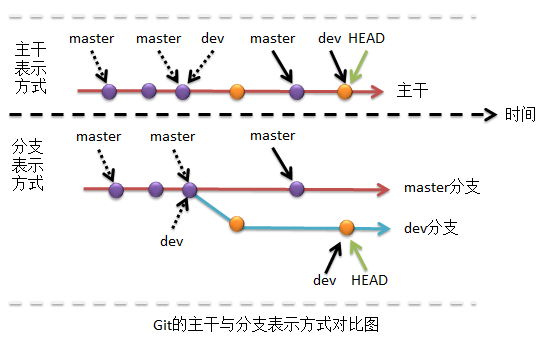
#### Git主干



Git是以时间为主线对版本进行管理的，而这条时间主线就是Git的主干。主干上的每个节点就是一个版本，即一次commit提交。在主干上可以定义多个指针，指向不同的节点。Git默认会创建一个名称为master的指针。

默认情况下用户操作的是master指针，但用户可以通过命令对操作的指针进行切换。用户每提交一次，就会形成一个新的节点，当前操作的指针就会向前移动一次。

#### Git分支



为了更加形象的描述指针的移动轨迹，我们称某一个指针的移动轨迹为一个分支。这样的话，一个指针就代表了一个分支，可以使用不同的指针操作不同的分支。master指针的移动轨迹与Git的主干重合，所以称为master主分支。

Git中还有一个特殊的指针叫HEAD，代表当前版本。其总是指向当前分支的当前版本，即总是指向当前分支指针所指向的节点（版本）。

#### 分支操作

对于分支，我们可以进行创建、切换、合并，及删除操作。

##### 分支创建

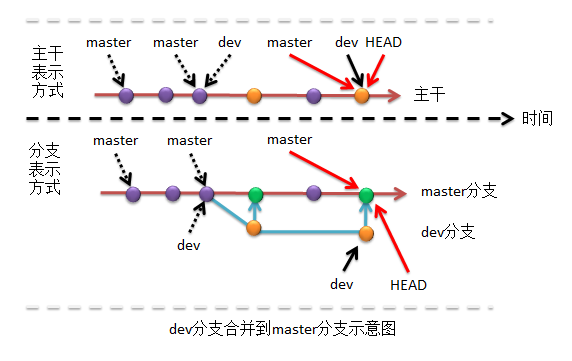
Git创建一个分支很快，例如，创建一个新的分支dev，就是创建一个dev指针。

##### 分支切换

分支的切换就是修改HEAD指针的指向。一旦切换到另一个分支dev，那么再往后的commit都是提交到了新分支dev上了，即dev指针向后移动，但master指针不动。

当然，在多分支下，可以在各个分支间任意切换，切换后均可进行各自的提交，而各自的提交与其它分支是无关的。

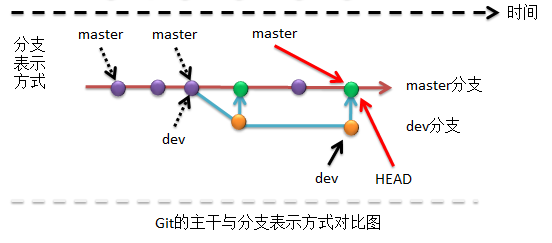
##### 分支合并



当dev分支上的工作完成后，需要将其合并到主分支master上。合并过程也很快，只需要将master指针指向dev指针指向的节点，然后再将HEAD指针指向master指针指向的节点即可。即分支的合并就是修改了两个指针的指向而已。

对于合并的较形象的理解是，合并就是将原来在dev分支上的节点，全部投射到master分支上，即全部合并到master分支上。或者说，将dev分支合并到master分支，就是将dev分支上的节点纳入到master分支上，归master分支管理。

##### 分支删除



对于分支的删除需要注意，必须在其它分支上才可删除另一分支，即分支不可“自销毁”。其实分支删除，删除的仅仅就是分支指针，而该分支上曾经的节点是不会被删除的。一般情况下，只有在将某分支合并到其它分支后才可将该分支删除。

### 分支基本操作

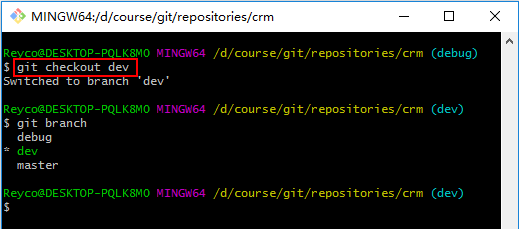
#### 创建分支git branch <branch>

该命令仅仅就是新建一个指定分支，但并不会切换到该分支。

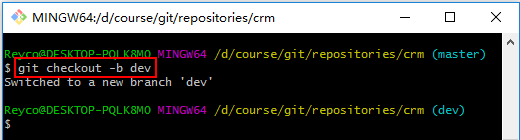
#### 查看系统分支git branch



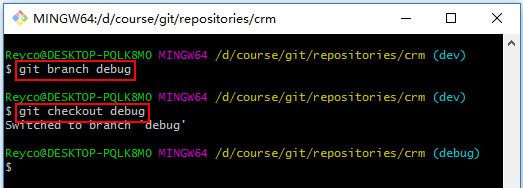
#### 切换分支git checkout <branch>



#### 创建并切换分支git checkout -b <branch>



该命令等价于两条命令：



#### 删除分支git branch –d <branch>



#### 强制删除分支git branch –D <branch>

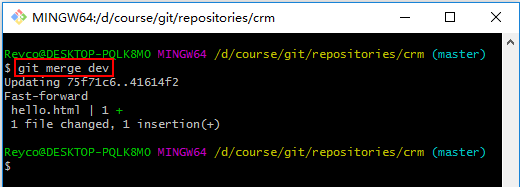
该命令无需合并即可删除。

### 分支合并与冲突

#### 合并分支git merge <branch>

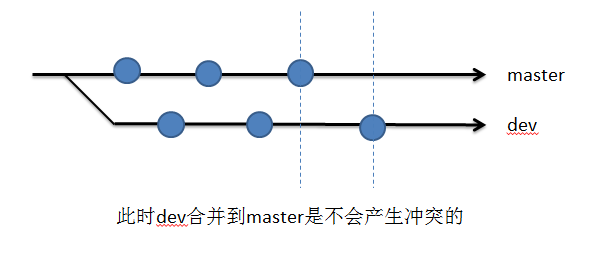
对于分支的合并需要注意，若要将分支B合并到分支A上，首先要切换到A分支上，然后再运行合并命令。

将dev分支合并到master分支。要确保当前处于master分支。

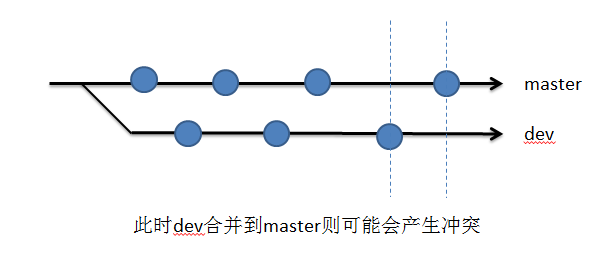


#### 合并冲突在什么情况下会产生？

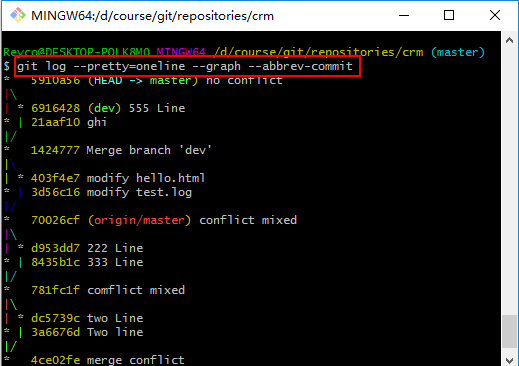
例如，现在要将dev合并到master，但dev的内容更新在时间主干上是最后一个。此时将dev合并到master是不会产生冲突的。



若master的内容更新在时间主干上是最后一个，且dev与master上最后一个提交版本内容不相同，此时将dev合并到master则可能会产生冲突。



#### 查看历史版本中的合并显示



# Git远程版本库

## 免密登录

Git的本地版本库要不时的从远程库上拉取数据及向远程库推送数据，但每次都要通过账号与密码的身份验证才可访问，非常麻烦。所以我们要设置免密登录，使Git的本地版本库可以免密登录并访问远程库。

### 免密登录机制

若主机间的通信采用的是SSH协议，即Secure Shell协议，那么可以设置免密登录。该协议的免密登录机制要求主机间采用SSH-Key即SSH密钥进行身份验证。

#### 公钥与私钥

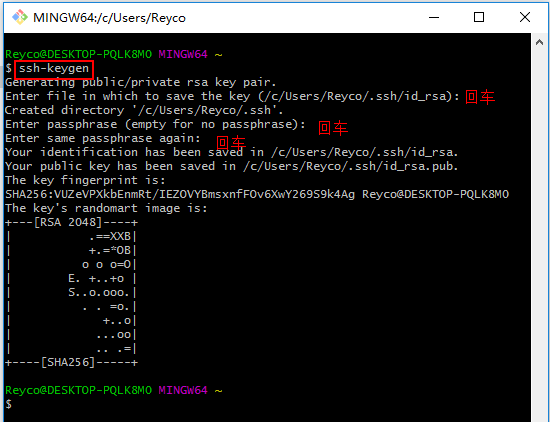
* “不对称加密方式”与“对称加密方式”
* 公钥与私钥

#### 免密登录工作原理

#### 本地生成密钥ssh-keygen

在本地主机运行ssh-keygen命令，生成SSH密钥。

在当前用户主目录下生成隐藏目录.ssh，其下会生成两个隐藏文件， id\_sra与id\_rsa.pub。其中id\_sra中存放的是私钥，id\_sra.pub中存放的是公钥。



## 码云Gitee远程版本库

### 码云简介

码云是阿里专为开发者提供稳定、高效、安全的云端软件开发协作平台，无论是个人、团队、或是企业，都能够用码云实现代码托管、项目管理、协作开发。码云同样也只支持Git 作为唯一的版本库格式进行托管。

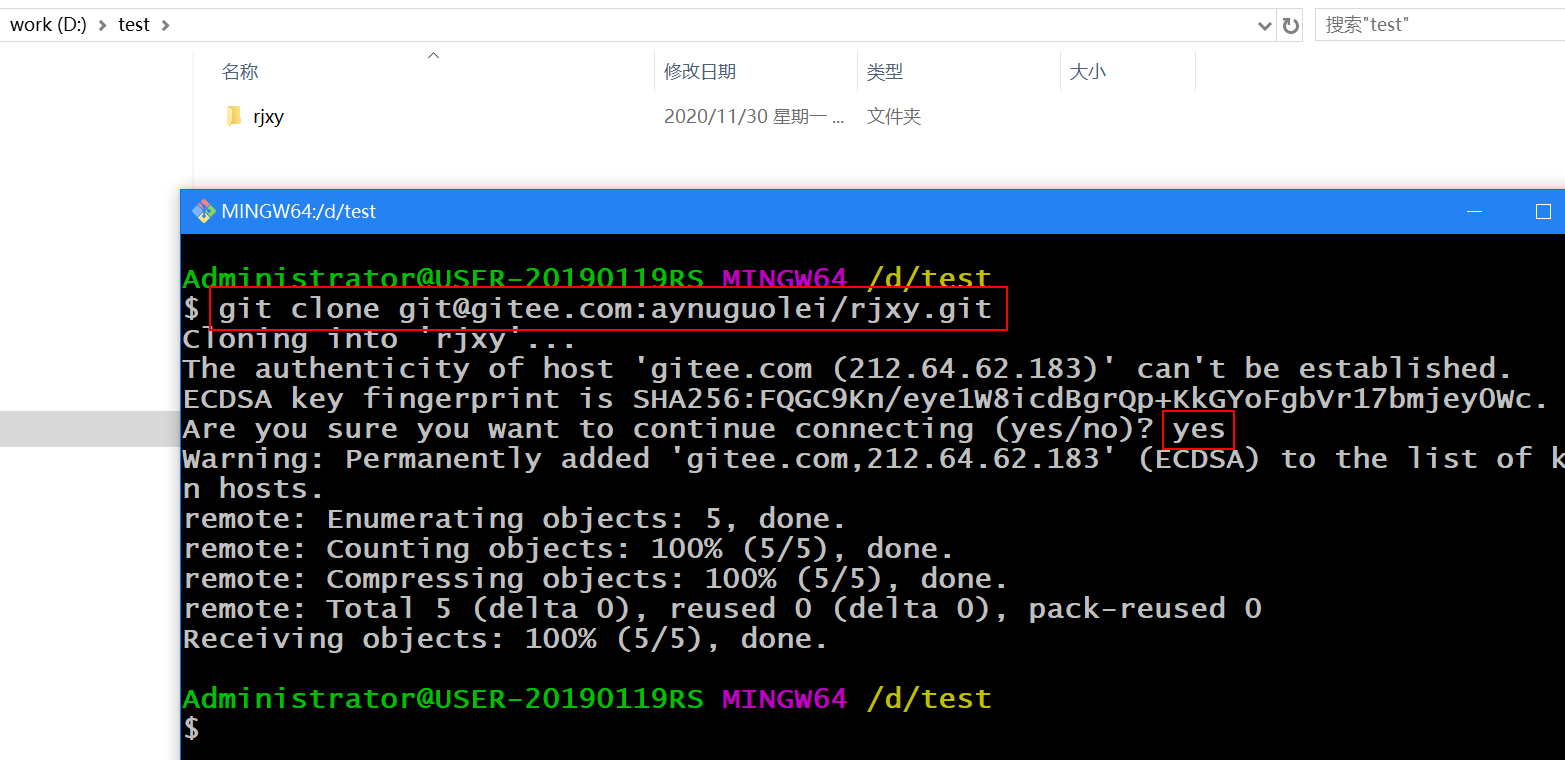
码云的免费版本也提供私有库功能，只是有5人的成员上限。

官网<http://gitee.com>。

### 本地库与远程库间的操作

#### 从远程库clone到本地库git clone

进入要存放远程库的目录，打开Git Bash，然后再运行克隆命令。



#### 修改本地库

在clone的本地库中新建一个文件hello.log，然后add、commit。

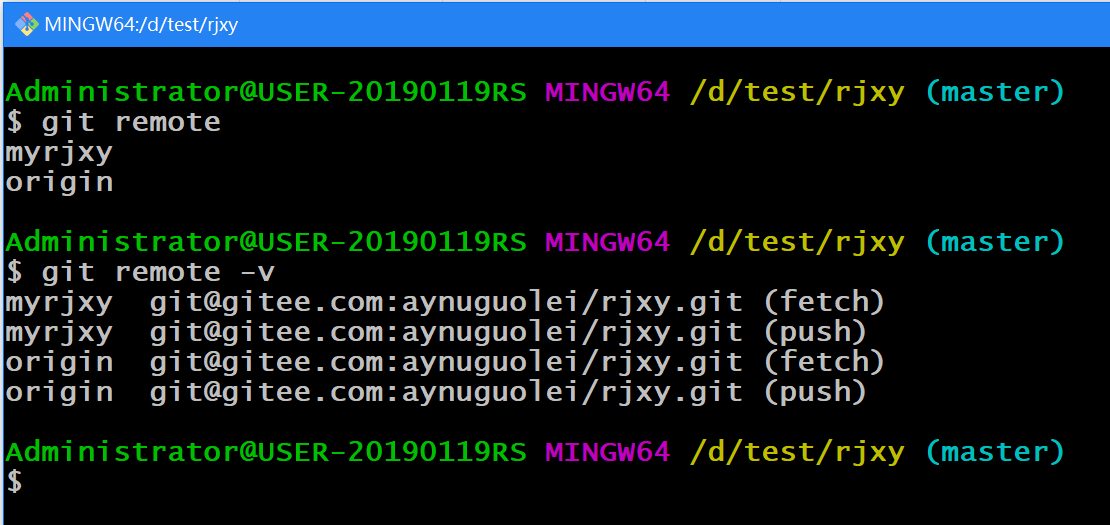
#### 命名远程库git remote add

为了方便后期对远程库的操作，我们会对远程库进行命名。

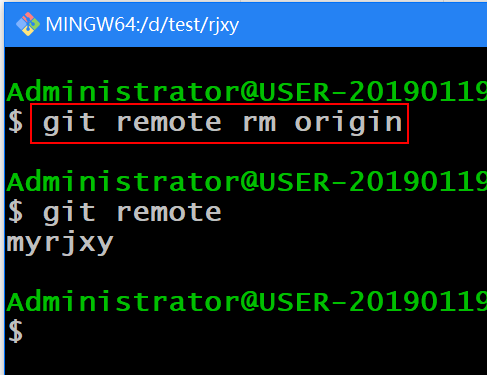


#### 查看本地的远程库信息git remote

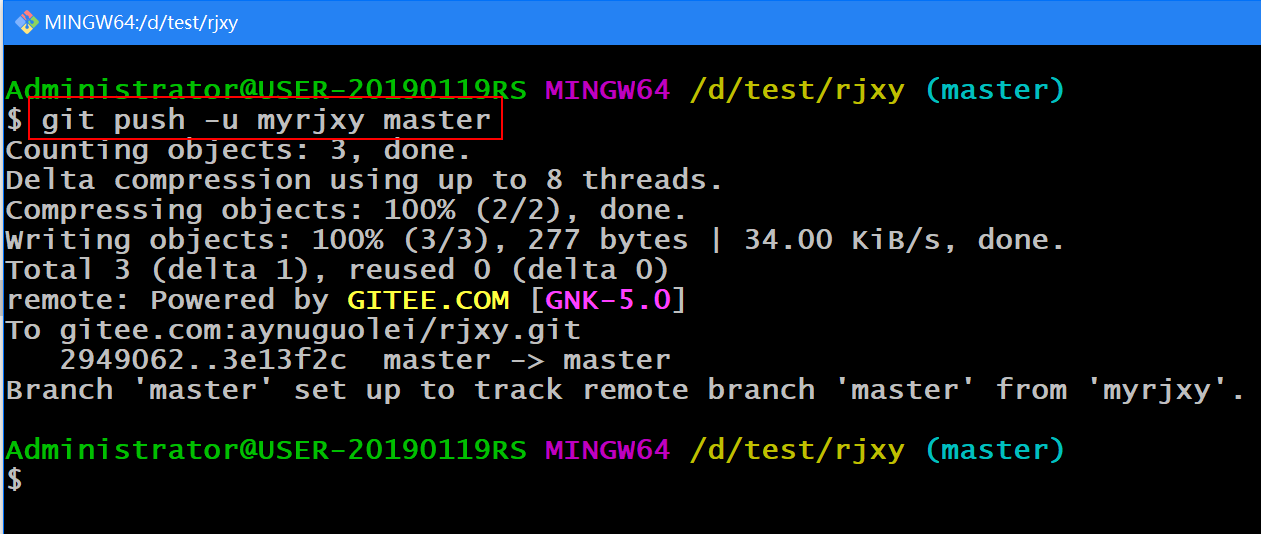
该命令可列举出当前本地版本库可操作的远程版本库名称，及更为详细的信息：远程库地址及本地库可执行的操作权限。



#### 删除本地的远程库信息git remote rm



#### 本地库内容推送到远程库git push



* git push myrjxy dev：将当前分支内容推送到远程库myrjxy的dev分支
* git push myrjxy：将当前分支内容推送到远程库myrjxy的master分支，不写分支，默认操作的是master分支
* git push –u myrjxy dev：将当前分支内容推送到远程库myrjxy的dev分支，并将myrjxy远程库作为默认操作的远程库。即下面的命令与该命令等价git push dev。同样的道理，下面的命令操作的是myrixy远程库的master分支 git push

#### 修改远程库中的文件

在远程库中直接修改hello.log文件内容。

#### 从远程库拉取到本地库git pull

将远程库中的内容pull到本地库中。

