Git

Zhao Xue

2017/04/21

# 初识 Git

### 工作流

本地仓库由git维护的三棵"树"组成：

* 第一个是 **Working Directory** ,它存储你的实际文件;
* 第二个是 **Index / Strage** ,类似于缓存区域,临时保存你的改动;
* 第三个是 **HEAD** ,它指向你最后一次提交的结果.

### 创建新仓库

创建或者切换到想要进行版本控制的文件夹,然后执行**git init**

### 克隆远端服务器github上的仓库

* git clone <https://github.com/zhaoxue-xmu/RDA.git>

指定创建的本地仓库的名称：

* git clone <https://github.com/zhaoxue-xmu/RDA.git> MyRDA

# 基本命令和操作

### 添加和提交

首先，在本地仓库中创建和编辑一个first.txt文件

* git add first.txt #添加到暂存区 git add \*
* git commit -m 'add first.txt' #提交改动到HEAD

### 查看仓库中文件的状态

* git status #提示未被跟踪的文件和已经修改的文件
* git log #查看之前提交的历史记录

## 移除文件

当不想对某个文件进行版本控制的时候，就需要把它从git中移除； 从git中移除一个文件，本质上做的事情就是把这个文件从暂存区删除，然后提交； 如果直接从文件夹中删除该文件，git依然会跟踪这个文件，并将一直提示这个文件的删除状态。

**用git rm 命令把文件从暂存区删除**：

* git rm --cached file

**删除源文件**：

* git rm first.txt.tmp

**提交**：

* git commit -m 'remove file'

**查看**：

* git log #可以查到remove file这个状态

## 推送改动

本地git init创建的仓库，git remote不会看到任何信息，因为没有添加任何远程仓库； 当你从远程仓库clone一个仓库到本地时，会自动将此地址加到你的远程仓库列表中,并默认命名为origin。

**添加一个远程仓库**:

* git remote add [shortname][url]
* git remote add myremote [git@github.com](mailto:git@github.com):zhaoxue-xmu/how-to-use-git.git
* git remote #查看对应的远程仓库myremote
* git remote -v #可以看到更加详细地信息(克隆的源地址)

**删除某个远程仓库**:

* git remote rm myremote #将myremote这个远程仓库从列表中清除

### 推送

* git push [remote-name][branch-name] #仓库 + 分支
* git push myremote master

## 分支

分支是用来将特性开发绝缘开来的，在你创建仓库的时候，master是默认的分支。在其他分支上进行开发，完成后再将它们合并到主分支上。

**创建分支**:

* git branch bracnch-name
* git log --decorate #查看当前提交和指针指向的分支(引用)

**切换分支**:

* git checkout bracnch-name
* git log --decorate --oneline --graph --all #--graph 以图形化的形式展示主、分支

**分支的删除**:

* git branch -d bracnch-name
* git log --decorate --all --graph --oneline - - - -

### 更新

* git pull #以在你的工作目录中获取(fetch)并合并(merge)远端的改动.
* git pull myremote master

这样会把远程仓库抓取到本地，再将远端的master分支合并到本地的master分支上。所以当你之前clone了代码，而其他开发者又更新了远端的代码后，可以通过git pull将本地更新到最新的版本。

### 不同的仓库时：

* git pull origin master --allow-unrelated-histories

## 合并

首先切换到主分支，然后执行:

* git merge branch-name

在上面两种情况下，git都会尝试去自动合并改动，但并非每次都能成功，可能出现冲突，这时候需要手动修改文件来解决冲突，然后添加和提交冲突。

* git diff #合并改动之前，查看差异

## 标签

为软件发布创建标签是推荐的，这个概念早已存在，在SVN中也有。 执行如下命令创建一个叫做1.0.0的标签：

* git tag 1.0.0 1b2e1d63ff

Note: 1b2e1d63ff是想要标记的提交ID的前10位字符，可以使用git log 命令获取提交的ID，也可以使用少几位地ID前几位，只要该ID具有唯一性。

## log

如果查看本地仓库的历史记录,最简单的命令就是git log，你也可以添加一些参数来修改输出。

**查看某一的人的提交记录**：

* git log --author=bob

**一个压缩后的每一条提交记录只占一行的输出**：

* git log --pretty=oneline

**通过ACSII艺术的树形结构来展示所有的分支,每个分支都标示名字和标签**：

* git log --graph --oneline --decorate --all

**查看哪些文件改变了**：

* git log --name-status

更多信息，参考git log --help

# END