# Git使用学习

1. Git是什么？

Git是目前世界上最先进的分布式版本控制系统

1. 集中式版本控制系统：

版本库是集中存放在中央服务器的，而干活的时候，用的都是自己的电脑，所以要先从中央服务器取得最新的版本，然后开始工作

1. 分布式版本控制系统：

首先，分布式版本控制系统根本没有“中央服务器”，每个人的电脑上都是一个完整的版本库

1. 比较：

集中式版本控制系统相比，分布式版本控制系统的安全性要高很多，因为每个人电脑里都有完整的版本库，某一个人的电脑坏掉了不要紧，随便从其他人那里复制一个就可以了

1. 优点：

分布式版本控制系统通常也有一台充当“中央服务器”的电脑，但这个服务器的作用仅仅是用来方便“交换”大家的修改，没有它大家也一样干活，只是交换修改不方便而已

1. 集中式版本库控制系统有哪些：

CVS，SVN，ClearCase，VSS，BitKeeper

1. 安装Git

顺序安装完git之后执行命令：设置参数

$ git config --global user.name "Your Name"$ git config --global user.email "email@example.com"

git config命令的--global参数，用了这个参数，表示你这台机器上所有的Git仓库都会使用这个配置，当然也可以对某个仓库指定不同的用户名和Email地址。

1. 理解:

成一个目录，这个目录里面的所有文件都可以被Git管理起来，每个文件的修改、删除，Git都能跟踪，以便任何时刻都可以追踪历史，或者在将来某个时刻可以“还原”。

1. pwd命令

pwd命令用于显示当前目录，例如仓库位于/Users/michael/learngit。

1. git init命令

把这个目录变成Git可以管理的仓库，Git就把仓库建好了，而且告诉你是一个空的仓库（empty Git repository），细心的读者可以发现当前目录下多了一个.git的目录，这个目录是Git来跟踪管理版本库的，没事千万不要手动修改这个目录里面的文件，不然改乱了，就把Git仓库给破坏了。如果你没有看到.git目录，那是因为这个目录默认是隐藏的，用ls -ah命令就可以看见。

1. 步骤

（1）把文件放到learngit目录下

（2)使用git add命令

（3)然后使用git commit -m “内容”

注：因为commit一次可以提交很多文件，所以add可以添加很多个文件

(1)git add <file>

(2)git commit -m <message>

$ git add readme.txt

$ git commit -m "append GPL"

第一步是用git add把文件添加进去，实际上就是把文件修改添加到暂存区；

第二步是用git commit提交更改，实际上就是把暂存区的所有内容提交到当前分支。

1. 查看加载到Git中的文件内容命令

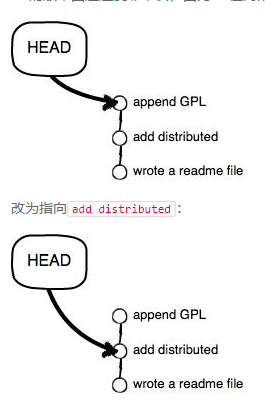
$ cat readme.txt

1. 查看文件更新日志

$ git log --pretty=oneline

1. Git版本退回或前进的工作本质

就是Head指针指向的位置问题，如果向前就更改Head指针的位置



1. 文件回退命令

$ git reset --hard HEAD^

$ git reset --hard commit\_id

1. 删除文件

$ rm test.txt

$ git rm test.txt

$ git **commit** -m "remove test.txt"

另一种情况是删错了，因为版本库里还有呢，所以可以很轻松地把误删的文件恢复到最新版本：

$ git checkout -- test.txt

1. 添加远程仓库

$ git remote add origin git@github.com:zhou0726a/learngit.git

1. 推送内容到远程

用git push命令，实际上是把当前分支master推送到远程。

第一次推送时：加-u $ git push -u origin master

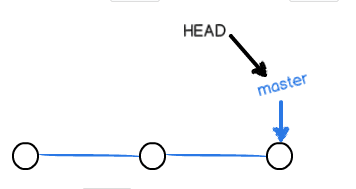
1. 把本地master分支的最新修改推送至GitHub

$ git push origin master

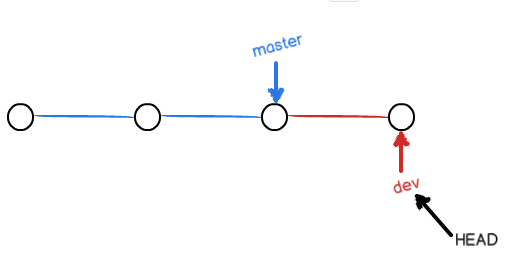
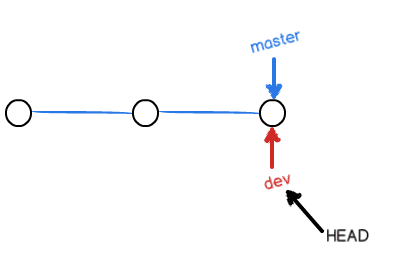
1. 从GitHub获取克隆一个本地库

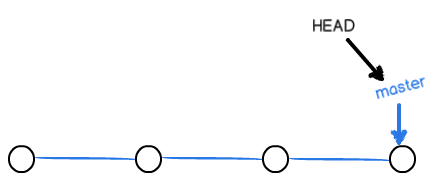
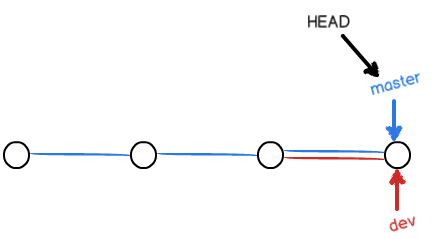
$ git clone git@github.com:zhou0726a/gitskills.git

1. 分支

每次提交，Git都把它们串成一条时间线，这条时间线就是一个分支。截止到目前，只有一条时间线，在Git里，这个分支叫主分支，即master分支。HEAD严格来说不是指向提交，而是指向master，master才是指向提交的，所以，HEAD指向的就是当前分支。

每次提交，master分支都会向前移动一步，这样，随着你不断提交，master分支的线也越来越长：





1. 创建dev分支，然后切换到dev分支

$ git checkout -b dev

等同于：$ git branch dev $ git checkout dev

1. $ git branchgit branch命令查看当前分支
2. 查看分支：git branch
3. 创建分支：git branch <name>
4. 切换分支：git checkout <name>
5. 创建+切换分支：git checkout -b <name>
6. 合并某分支到当前分支：git merge <name>
7. 删除分支：git branch -d <name>
8. 解决冲突
9. 创建一个分支

git checkout -b feature1

1. 在feature1分支上提交

git add readme.txt git commit -b “XXXXXX”

1. 调转到master分支上
2. 在master分支上修改提交，会出错，有冲突
3. 视图合并

git merge feature有冲突，手动解决然后再提交

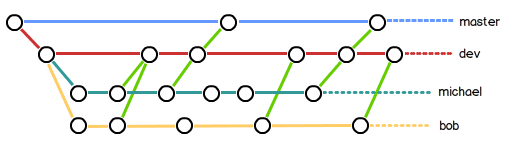
1. 最后删除feature1分支

有冲突时，先手动解决，再提交。

1. 分支管理

$ git merge --no-ff -m "merge with no-ff" dev

Fast forward模式下删除分支后会丢掉分支信息，--no-ff方式下git merge 不会有此类问题。



1. Bug分支

问题：遇到Bug时，需要停掉手中的工作，创建一个分支取处理Bug

解决：git stash把当前工作先储藏起来，等以后可以继续工作，修复Bug后提交，然后切换成master分支，并完成合并，最后删除修改Bug的分支即可。

$ git merge --no-ff -m "merged bug fix 101" issue-101

恢复：

Git stash apply

删除：

git stash drop

恢复+删除：

git stash pop

检查：

$ git stash list

1. Feature分支

当开发一个新功能时，开发完之后需要撤回删除,此时feature-vulcan还没有被合并

git branch -d feature-vulcan

但是需要强行删除

$ git branch -D feature-vulcan

多人协作

查看远程库的信息：

$ git remote

查看详细信息：

$ git remote -v

推送分支：

$ git push origin master

推送其他分支：

$ git push origin dev

两人同时推送时发生错误，需要先在本地

$ git push origin dev

$ git pull

合并有冲突后需要手动解决错误，然后进行提交

$ git push origin dev

多人协作模式

1. 首先，试图可以用git push origin <branch-name>推送自己的修改;
2. 如果推送失败，则因为远程分支比你的本地更新，先需要用git pull试图合并;
3. 如果合并有冲突，则解决冲突，并在本地提交;
4. 没有冲突或者解决掉冲突后，再用git push origin <branch-name>推送就能成功！
5. 如果git pull提示no tracking information，则说明本地分支和远程分支的链接关系没有创建，用命令git branch --set-upstream-to <branch-name> origin/<branch-name>。