

Git学习报告

报告人：郭峻宏 | 时间： 2022/1/6

# Git简介：

1. 简介：

分布式控制系统，与Linux操作系统共生，与CVS以及SVN等集中式控制系统对立。

1. 优点：
2. 不需要联网。
3. 分支管理，并行操作。
4. 快照式保存历史纪录。

# Git内容学习：

## 创建版本库：

1. 用mkdir创建一个文本。
2. 用git init将其初始化为Git可管理仓库。
3. 自己创建一个文件（最好是使用VScode），将文件移动到第一步创建的文本下。
4. 用git add和git commit -m “name”将文件提交到仓库。

注：git commit -m可以同时提交多个文件。

## 文件历史回溯：

代码补充：

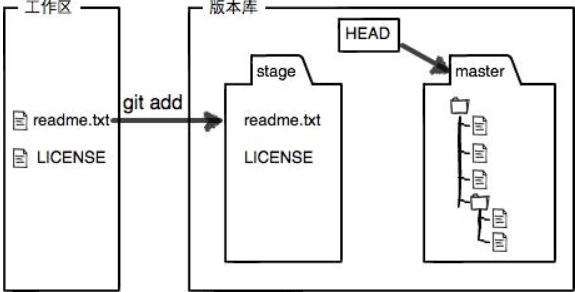
1. git status可以掌握工作区状态。
2. git diff可以查看修改内容。
3. cat可以查看文本内容。

#### 版本回退：

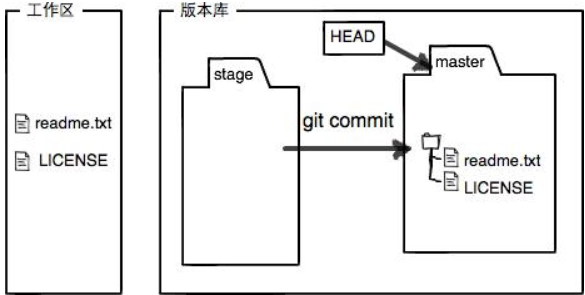
1. 用git log查询提交历史记录（用git log –pretty=oneline可以整理输出历史信息）。
2. 用git reset –hard commit\_id跳跃不同历史版本，其中HEAD表示当前版本，HEAD^表示上一历史版本，多版本可用HEAD~，也可用版本号跳跃（输入前几位就行）。
3. 用git reflog可以查看命令历史，找到历史版本号。

#### 工作区和暂存区：

简介：暂存区暂存了用git add指令添加进去的文件（可以存放多个文件），用git commit -m “name”可以将缓存区所有文件存入工作区文本下。



图一：add功能执行



图二：commit功能执行

#### 撤销修改：

用git checkout – filename，可以舍弃修改，即回到最近一次git commit或git add时的状态。

1. 文件自修改后还没有被放到暂存区，撤销修改就回到和版本库一模一样的状。
2. 文件已经添加到暂存区后，又作了修改，现在，撤销修改就回到添加到暂存区后的状态。再用git reset HEAD filename可以把缓存区的修改撤销，重新放回工作区，然后再用git checkout撤销工作区的修改，撤销修改就回到原状。
3. 文件已经提交到了版本库，可用版本回溯到之前版本。

#### 删除文件：

1. 在文件管理库中直接删除，然后使用git rm filename和git add filename，即可删除或复原文件
2. 使用命令行指令：
3. 用rm filename删除文件。
4. 选择：的确要删除，则用git rm filename后，再用git commit -m “remove filename“，就可执行删除。如果删除错误，可用git checkout – filename恢复删除文件。

注：从来没有被添加到版本库就被删除的文件无法恢复。

## 远程仓库：

#### 添加与删除远程库：

1. 添加远程库：
2. 在Github上创建一个新的仓库。
3. 在本地仓库下运行git remote add origin git@github.com:GitHub用户名/仓库名.git（远程库名字默认为origin）。
4. 用git push -u origin master将本地库内容传输到远程库（第一次传输使用-u可以将本地的master分支和远程的master分支连接起来，以后可简化命令为git push origin master推送最新修改）。
5. 用git push origin master将本地master分支最新修改推送到GitHub。
6. 删除远程库：
7. 用git remote –v查看远程库信息。
8. 根据名称，用git remote rm name删除（删除了绑定关系）。
9. 登录GitHub，后台删除。

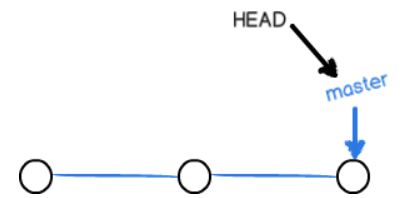
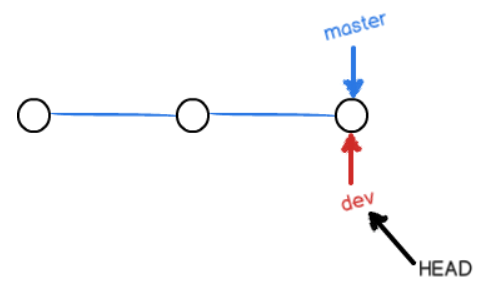
注：有无联网都可正常操作。

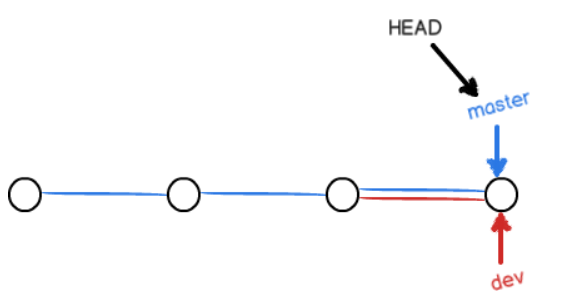
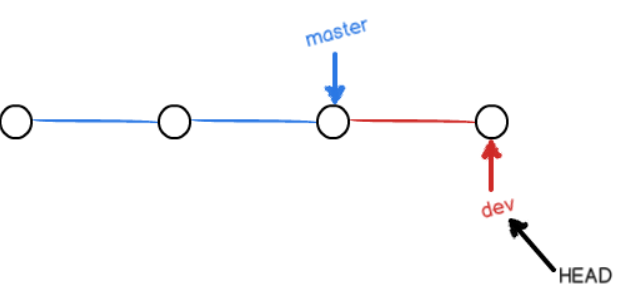
#### 从远程库克隆：

1. 创建README.md文件。
2. 用git clone git@github.com:Github用户名/文件名.git即可克隆一份文件到本地文本下。

## 分支管理：

#### 创建与合并分支：

图三：分支创建于合并示意图

1. 创建分支：
2. 用git branch创建分支。
3. 用git checkout切换分支（两行代码相当于git checkout -b）。
4. 用git branch查看当前分支（当前分支前会有标号\*）。
5. 对文件进行操作。
6. 用git checkout切回主分支。
7. 合并分支：
8. 用git merge和并指定分支到当前分支。
9. 用git branch -d删除分支。

注：可用git switch -c来创建并切换到新分支。

#### 解决冲突：

当在分支在主分支中均对文件进行修改，合并时会产生冲突，这时我们需要手动编辑合并失败的文件在提交，合并完成。

用git log –graph可以查看分支合并图。

#### 分支管理策略：

用Fast forward模式在删除分支后，会丢掉分支信息，可用git merge –no-ff -m “commit\_id” branchname来合并分支，这样可以保留分支。

#### Bug分支：

1. 修改主分支bug：
2. 用git stash查看工作区，将工作现场储存起来。
3. 创建新分支，修改bug，合并分支，删除分支。
4. 用git stash apply恢复stash，再用git stash drop删除（或用git stash pop直接恢复删除）
5. 修改其他分支bug：
6. 可重复以上操作实现对其他分支同样的修改。
7. 用git cherry-pick来实现复制一份修改操作。

#### Feature分支：

Feature分支多用于开发。分支在未合并前，删除会丢失掉修改，这时用git branch -D branchname强行删除。

#### 多人协作：

1. 可以试图用git push origin branchname推送自己的修改。
2. 如果推送失败，需要先用git pull试图合并。
3. 如果合并有冲突，则手动解决冲突，并在本地提交。
4. 如果没有冲突或者冲突解决，再用git push origin branchname推送。

注：可用git remote -v擦看远程库信息。

#### Rebase：

用git rebase操作后再用git log查看，可以将原本的分支提交变成一条直线，使得我们在查看历史提交变化时更容易。

## 标签管理：

Git的标签虽然是版本库的快照，但其实它就是指向某个commit的指针（分支可以移动，标签不能移动），能够更加简洁的查找某个历史版本。

#### 创建标签：

1. 创建标签：
2. 用git checkout或者git switch切换到需要打标签的分支上。
3. 用git tag tagname给分支打上标签。
4. 用git tag可以查看所有标签。
5. 创建历史记录标签：
6. 用git log查找到需要打标签的commit\_id。
7. 用git tag tagname commit\_id为该分支打上标签。
8. 用git tag可以查看所有标签。

注：可以用git show tagname查看标签信息。可以用git -a tagname -m “注释” commit\_id创建带有说明文字的标签。

#### 操作标签：

1. 删除标签：
2. 删除本地标签：用git tag -d tadname删除。
3. 删除远程标签：现在本地用git tag -d tadname删除， 再在远程用git push origin :refs/tags/tagname删除。
4. 推送标签：
5. 推送某个标签：用git push origin tagname。
6. 推送全部标签：用git push origin –tags。