廖雪峰git教程笔记

集中式版本控制系统 vs 分布式版本控制系统

- 集中式版本控制系统：版本库集中存放在中央服务器，需要联网才能工作
- 分布式版本控制系统：每个人电脑上都是一个完整的版本库。
- 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，但是这个服务器的作用仅仅用来方便“交换”大家的修改

版本回退

- HEAD指向的版本就是当前版本，因此，Git允许我们在版本的历史之间穿梭，使用命令git reset --hard commit_id。
- 穿梭前，用git log可以查看提交历史，以便确定要回退到哪个版本。
- 要重返未来，用git reflog查看命令历史，以便确定要回到未来的哪个版本。

在Git中， 用*HEAD*表示当前版本

–hard参数的意义?

工作区----暂存区----分支

管理修改

```
Git管理的是修改，而不是文件
git checkout -- file可以丢弃工作区的修改
```

用命令git reset HEAD file可以把暂存区的修改撤销掉（unstage），重新放回工作区

```
场景1：当你改乱了工作区某个文件的内容，想直接丢弃工作区的修改时，用命令git checkout -- file。

场景2：当你不但改乱了工作区某个文件的内容，还添加到了暂存区时，想丢弃修改，分两步，第一步用命令gi

场景3：已经提交了不合适的修改到版本库时，想要撤销本次提交，参考版本回退一节，不过前提是没有推送到
```

远程仓库

要关联一个远程库，使用命令git remote add origin git@server-name:path/repo-name.git；

关联后，使用命令git push -u origin master第一次推送master分支的所有内容；

此后，每次本地提交后，只要有必要，就可以使用命令git push origin master推送最新修改；

创建分支

```
查看分支：git branch

创建分支：git branch <name>

切换分支：git checkout <name>
```

```
创建+切换分支：git checkout -b <name>

合并某分支到当前分支：git merge <name>

删除分支：git branch -d <name>
```

解决冲突

用git log --graph命令可以看到分支合并图。

分支管理策略

通常在合并分支的时候，如果可能Git会用Fast forward模式，但是，在这种模式下，删除分支后，会丢掉分支信息。

如果要强制禁用Fast forward模式，Git会在merge时生成一个新的commit，这样，从分支历史上就可以看出分支信息。

合并分支时，加上–no-ff参数就可以用普通模式合并，合并后的历史有分支，能看出来曾经做过合并，而fast forward合并就看不出来曾经做过合并。

Bug分支

修复bug时，我们会通过创建新的bug分支进行修复，然后合并，最后删除；

当手头工作没有完成时，先把工作现场git stash一下，然后去修复bug，修复后，再git stash pop，回到工作现场。

开发一个新feature，最好新建一个分支；

如果要丢弃一个没有被合并过的分支，可以通过git branch -D 强行删除。

多人协作

- 首先，可以试图用git push origin branch-name推送自己的修改；
- 如果推送失败，则因为远程分支比你的本地更新，需要先用git pull试图合并；
- 如果合并有冲突，则解决冲突，并在本地提交；
- 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！

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 branch-name origin/branch-name。

标签管理

发布一个版本的时候，我们通常在版本库中打一个标签，这样就唯一确定了打标签时刻的版本。将来无论什么时候，取某个标签的版本，就是把那个打标签的时刻的历史版本取出来。所以标签也是版本库的一个快照。

其本质是指向某个commit的指针

分支vs标签： 分支可以移动，标签不可以

为什么要引入tag：tag是一个更容易让人记住的有意义的名字，它跟某个commit绑在一起。

```
命令git tag <name>用于新建一个标签，默认为HEAD，也可以指定一个commit id；

git tag -a <tagname> -m "blablabla..."可以指定标签信息；

git tag -s <tagname> -m "blablabla..."可以用PGP签名标签；

命令git tag可以查看所有标签。
```

操作标签：

```
命令git push origin <tagname>可以推送一个本地标签；

命令git push origin --tags可以推送全部未推送过的本地标签；

命令git tag -d <tagname>可以删除一个本地标签；

命令git push origin :refs/tags/<tagname>可以删除一个远程标签。
```

忽略特殊文件

忽略某些文件时，需要编写.gitignore；

.gitignore文件本身要放到版本库里，并且可以对.gitignore做版本管理！

配置别名

```
git config --global alias.st status
git config --global alias.co checkout
git config --global alias.ci commit
git config --global alias.br branch
git config --global alias.unstage 'reset HEAD'
// 散心病狂？ nonono 一点都不
git config --global alias.lg "log --color --graph --pretty=format:'%Cred%h%Creset
```

Problem & Solution

Problem1:

```
refusing to merge unrelated histories
```

Solution:

```
You can use --allow-unrelated-histories to force the merge to happen.
```

Problem2:

```
The authenticity of host 'github.com (192.30.255.112)' can't be established.
```

Solution：

```
1    Are you sure you want to continue connecting (yes/no)? //输入yes，回车
```