Git相关知识

2020年6月25日 12:02

在本地有一个文件夹，这个文件夹一般被称为工作区，一般来说，一个仓库对应你本地的一个文件夹。

来自 <https://zhuanlan.zhihu.com/p/25624944>

查看用户：git config user.name

查看邮箱：git config user.email

来自 <https://blog.csdn.net/elephant230/article/details/90380408>

基本操作
git init 初始化一个仓库
git status 查看状态信息

来自 <https://zhuanlan.zhihu.com/p/25624944>

来自 <https://zhuanlan.zhihu.com/p/25624944>

- on branch master标记了你的分支（分支的概念之后说）
- untracked files说明这些文件只是在你的工作区做了修改，版本库并没有收到任何的回馈信息，这里会给出详细的提示

来自 <https://zhuanlan.zhihu.com/p/25624944>

git add demo.txt
git commit -m '*'

来自 <https://zhuanlan.zhihu.com/p/25624944>

add的作用是把这些推到暂存区
commit之后的-m参数是message，也就是说需要之后指定提交的信息，也就是引号里面的内容。添加这信息的目的是为了查找标记

来自 <https://zhuanlan.zhihu.com/p/25624944>

```
git log
git log --graph
git log --pretty=oneline
```

解释一下：

- 参数--graph用来标记分支的情况（合并等），如果只是一直的单分支，一般不需要这个参数
- --pretty=oneline，这个会使提交记录看着比较简洁

前面黄色的码是commit ID

来自 <https://zhuanlan.zhihu.com/p/25624944>

版本查找
git reflog查看修改过程

来自 <https://zhuanlan.zhihu.com/p/25624944>

git有两种思路进行回滚
git reset --hard *

--hard参数是为了把目前已经放到工作树（已经commit到版本库）上的信息都删除掉，以防回滚的时候产生冲突等不测。

1. *后面用可以使用HEAD^，回到上一步，如果你要回滚两步，就使用HEAD^^，当然了，一直往后面加这玩意加太多了也看着有点膈应，所以我们也可以使用HEAD~n表示回退到n步之前。
2. 也可以添加提交序列号（commit id），我们看到使用git reflog之后，前面有一个7位数的序列号，*修改为对应的记录的序列号，也可以做到回滚

来自 <https://zhuanlan.zhihu.com/p/25624944>

实际使用会遇到的问题及解决方法【始终保证远程库是最新版本】

本地上传文件到版本库
（可以在本地新建一个 文件夹，git init 初始化一个仓库，git status 查看状态信息），也可以在已有的工作区文件夹里，
git add demo.txt
git commit -m '*'

修改后不满意，想回滚到之前的版本
git log 查看提交日志（提交历史）
git reflog查看修改过程（命令历史）
git log --pretty=oneline 简洁版本的输出

git reset --hard *

1. *后面用可以使用HEAD^，回到上一步，如果你要回滚两步，就使用HEAD^^，当然了，一直往后面加这玩意加太多了也看着有点膈应，所以我们也可以使用HEAD~n表示回退到n步之前。
2. 也可以添加提交序列号（commit id），我们看到使用git reflog之后，前面有一个7

位数的序列号，*修改为对应的记录的序列号，也可以做到回滚

? 修改旧版本后悔了，想恢复到新版本
使用git reflog 查看修改历史，回到该一步骤

? 删除文件
删除后只需要commit
如果是误删，
git checkout -- test.txt
用版本库里的版本替换工作区的版本，无论工作区是修改还是删除，都可以“一键还原”。

? 从版本库上传到远程库
每使用一台新的设备，需要在github上加入新的SSH，设置好后，
ssh -T git@github.com
测试一下

git remote add origin git@github.com:*
git push -u origin master

- 第一行命令是要添加一个远程库，*后面是这个仓库的地址，对照好GitHub界面左上方仓库的地址，然后不要忘了后面的.git扩展名，remote 遥远的，origin一般表示远程库。
- 第二行命令是将master分支上的东西都推送到这个远程库上。-u参数表示第一次需要先建立本地版本库与GitHub远程库的连接，第二次之后就不需要这个参数了。

? 如何解决冲突（远程库和版本库的版本不一致）

```
$ git pull
There is no tracking information for the current branch.
Please specify which branch you want to merge with.
See git-pull(1) for details.

    git pull <remote> <branch>

If you wish to set tracking information for this branch you can do so with:

    git branch --set-upstream-to=origin/<branch> dev
```

`git pull` 也失败了，原因是没有指定本地 `dev` 分支与远程 `origin/dev` 分支的链接，根据提示，设置 `dev` 和 `origin/dev` 的链接：

```
$ git branch --set-upstream-to=origin/dev dev
Branch 'dev' set up to track remote branch 'dev' from 'origin'.
```

如果提示：fatal: refusing to merge unrelated histories

git pull --allow-unrelated-histories
pull成功后再将本地的修改push到远程库【始终保持远程库为最新版本】

? 从远程库下载到另一台本机

git clone git@github.com:* （会默认下载一个文件夹）

git remote -v 查看远程库信息

? 什么是分支branch

Git鼓励大量使用分支：

查看分支：`git branch`

创建分支：`git branch <name>`

切换分支：`git checkout <name>`或者`git switch <name>`

创建+切换分支：`git checkout -b <name>`或者`git switch -c <name>`

合并某分支到当前分支：`git merge <name>`

删除分支：`git branch -d <name>`