GIT

2021年4月20日 15:07

创建版本库

首先，选择一个合适的位置，创建一个空目录(learngit)

```
$ mkdir learngit
$ cd learngit
$ pwd
```

pwd 命令用于显示当前目录

第二步通过git init 命令把这个目录变成Git可以管理的仓库

```
$ git init
Initialized empty Git repository in
/Users/michael/learngit/.git/
```

自报家门

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

第三步添加文件到Git仓库

初始化仓库 使用 git init 命令

创建一个文件放到learngit目录或者子目录后

（1）用命令git add 告诉 Git,把文件添加到仓库

```
$ git add readme.txt
```

（2）用命令git commit 告诉Git ,把仓库提交到仓库

```
$ git commit -m "wrote a readme file"
```

为什么Git添加文件需要add，commit一共两步呢？因为commit可以一次提交很多文件，所以你可以多次add不同的文件。

版本退回

git log 命令显示从最近到最远的提交日志

```
$ git log
```

如果嫌输出信息太多，看得眼花缭乱的，可以试试加上--pretty=oneline

```
$ git log --pretty=oneline
```

把当前版本 回退到上以版本

```
$ git reset --hard HEAD^
HEAD is now at e475afc add distributed
```

查看readme.txt的内容是不是版本add distributed

```
$ cat readme.txt
```

git log 重现版本状态

最新的那个版本 `append GPL` 已经看不到了！好比你从21世纪坐时光穿梭机来到了19世纪，想再回去已经回不去了，肿么办？

办法其实还是有的，只要上面的命令行窗口还没有被关掉，你就可以顺着往上找啊找啊，找到那个 `append GPL` 的 `commit id` 是 `1094adb...`，于是就可以指定回到未来的某个版本：

```
$ git reset --hard 1094a
HEAD is now at 83b0afe append GPL
```

撤销修改

$ git restore readme.txt 可以丢弃工作区中的修改

这里有两种情况：

一种是readme.txt自修改后还没有被放到暂存区，现在，撤销修改就回到和版本库一模一样的状态；

一种是readme.txt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。

总之，就是让这个文件回到最近一次git commit或git add时的状态。

git reset HEAD <file>可以把暂存区修改重新放回工作区

删除文件

rm <test.txt> rm可以删除工作区中的文件，但是并没有删除版本库中的文件

若想删除版本库中的文件需要两步

```
$ git rm test.txt
rm 'test.txt'
$ git commit -m "remove test.txt"
```

上传到远程库

```
& git push origin master
```

创建分支

```
$ git checkout -b dev
```

加上-b参数表示创建并切换，相当于以下两条命令

创建

```
$ git branch dev
```

切换 建议用后面的 git switch <dev>

```
$ git checkout dev
```

查看当前分支：

```
$ git branch
```

然后可以在分支上正常提交，比如对 readme.txt做个修改，加上一行：Creating a new branch is quick.

```
$ git add read.txt
$ git commit -m "branch test"
```

当dev分支完成之后，就可以切换回master 分支：

```
& git checkout master
```

将dev 分支的工作成果合并到master分支上:

```
git merge dev
```

删除dev分支：

```
$ git branch -d dev
```

切换分支：

```
git switch master
```

查看文件内容

```
cat <readme>
```

BUG管理

把工作现场储存起来

```
$ git stash
```

查看 工作现场储存的位置

```
$ git stash list
```

恢复工作场所

方法1 恢复后stash未被删除

```
$ git stash apply
```

方法2 回复后stash被删除

```
$ git stash pop
```

同样的bug 在分支上也修改

```
$ git cherry-pick 4c805e2
```

或

Git checkout <branch>

显示所有分支的信息

Git branch -v