

Git学习笔记-参考廖雪峰老师

笔记本: [05]编程

创建时间: 2018/1/25 9:50

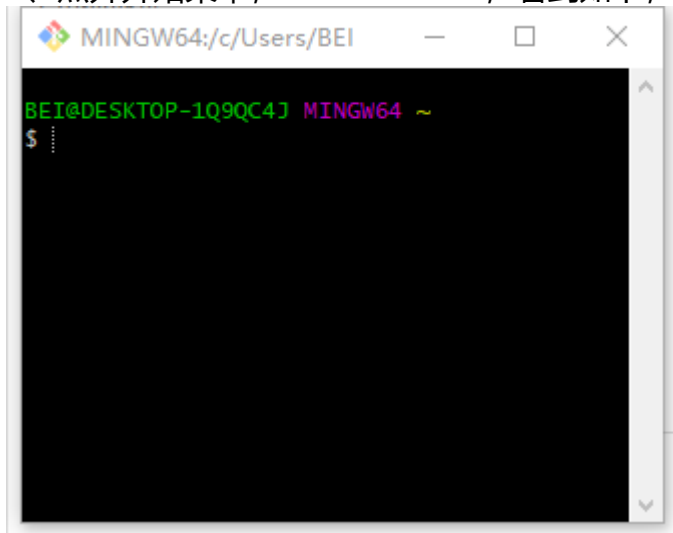
更新时间: 2018/1/30 12:31

URL: <https://www.liaoxuefeng.com/>

Windows为例, Linux基本一致。

20180125 安装Git

- 1、在windows上安装, 下载安装包, 默认安装选项。
- 2、点开开始菜单, Git→Git Bash, 看到如下, 说明安装成功。



- 3、设置名字和邮箱 (笔记本), 如下:

在我的台式机上设置如下

0125 创建第一个版本库

学习总结两点:

- 1、在一个文件夹内创建“仓库”, 首先命令行到达该文件夹下, 然后命令:

```
git init
```

- 2、添加文件到“仓库”, 两步走:

```
git add  
git commit -m "what note"
```

其中, commit的意思是“把.....托付给”。

我的操作：如下操作时，出现错误：

```
$ git add readme.txt
fatal:pathspec 'readme.txt' did not match any files
```

找出原因：

创建readme.txt时，我创建的名字就是"readme.txt"，但是，windows自动补齐扩展名。于是变成了"readme.txt.txt"，所以就报错找不到文件了。

0125 时光机穿梭-简介

假如两周前我修改了readme.txt文件中的内容，但是现在我不确定是否修改了，更不确定修改了什么。如何查看？

两个新命令

```
git status //查看仓库内的文件是否被修改，随时掌握工作区的状态
git diff //查看文件修改的内容
```

确定修改的内容是对的，接下来可以配合之前的命令一起使用，来进行版本更新

```
git add
git commit -m "what note"
```

当提交完毕后，再尝试查看status，可以发现工作目录是干净（working directory clean）。

0125 时光机-版本回退

1、版本回退的速度非常快，Git内部有个指向当前版本的指针，即HEAD。改变版本时，只是改变了HEAD的指向。

2、版本切换命令

```
git reset --hard commit_id
```

3、如何查看commit_id呢？使用命令

```
git log
```

简化内容，增加附属命令

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

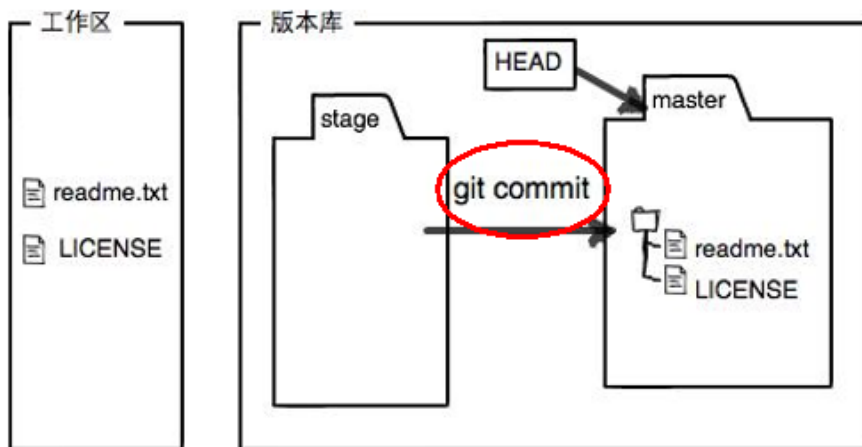
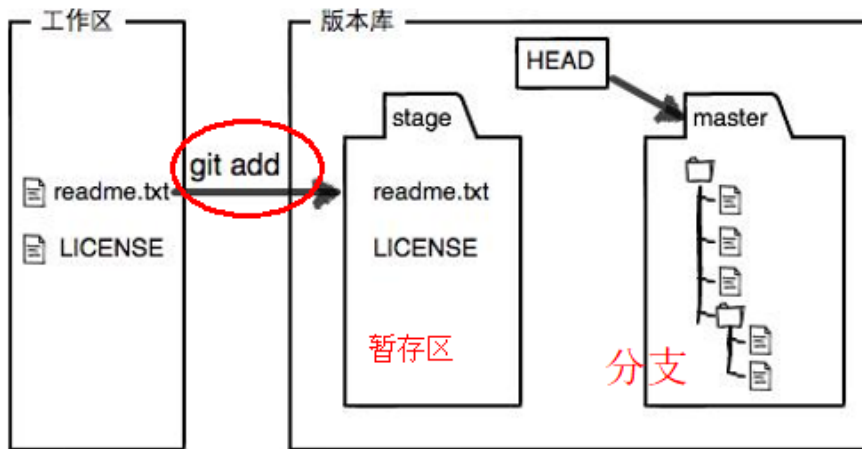
4、使用git log只能查看当前版本及更早的版本号，如果我穿梭到了之前的版本，然后想回来却找不到版本号了，怎么办？使用下面的命令，可以找到所有的版本号。

```
git reflog
```

0127 时光机-工作区和暂存区

在learngit文件夹中，有两部分：可见的一部分文件都是在工作区，".git"文件夹处于隐藏状态，里面是版本库。

- 1、工作区：可理解为“手术台”，和git版本库没关系。但是工作区的变化在git的监视下，可以用git status查看工作区的状态。
- 2、暂存区：版本库包括两部分，分支和暂存区。工作区的修改完毕后，将修改交给版本库时，先用git add命令加入暂存区，然后git commit放入某个分支的当前版本中。



- 3、使用git status查看工作区状态，一般有以下几个结果，其关键提示分别为：

Untracked files:

表示该部分文件从未添加过，需要git add（添加到暂存库）

Changes not staged for commit:

表示该部分文件已经添加过，但是再上一次commit之后又修改了，需要重新git add

```
Changes to be committed:
```

表示该部分文件都已经git add过了（已经在stage暂存库），需要git commit提交到分支的版本中

```
nothing to commit, working tree clean
```

表示工作区干净了，没啥要做的

0127 版本穿梭-管理修改

- 1、Git跟踪的是修改，而不是文件。所以，操作比较方便，“轻、快”。
- 2、Git add命令之后，需要git commit才能记录进入版本库。

0128 版本穿梭-撤销修改

- 1、如果改乱了工作区文件的内容（还没提交到stage），想直接丢弃工作区的修改，需要执行命令

```
git checkout -- <file>
```

注意--前后都有空格。

- 2、不但改乱了工作区文件的内容，还提交到了stage，然后想要撤销这些修改，需要执行两个命令。首先，将修改从stage中清退

```
git reset HEAD <file>
```

其次，使用1中的命令，在工作区中的文件里撤销这个修改。

- 3、做了一个修改，不但提交到了stage，并且commit过了，只能用版本回退了。详见“时光机-版本回退”一节。
- 4、不但stage和commit了，并且还推送到了远程版本库。此时搞不定了。

0128 版本穿梭-删除文件

- 1、在工作区误删一个文件，可以用下面的方式恢复。（文件中的内容只能恢复到最近一次的commit）

```
git checkout -- <file>
```

- 2、确实要删除一个文件，当在工作区删除后，还要执行以下命令

```
git rm <file>  
git commit -m "what note"
```

0128 远程仓库

1、本地生成git秘钥SSH文件，在c盘user文件夹的.ssh文件夹中。如果没有.ssh文件夹，则在Git Bash中运行命令

```
ssh-keygen -t rsa -C "myemail"
```

在.ssh文件夹中生成了id_rsa（私钥，不可公开）和id_rsa.pub（公钥，可以提交到github）两个文件。

2、建立Github账号，新建SSH秘钥，粘贴id_rsa.pub中的内容。

这样github网站就可以识别我的这台电脑了。

0128 远程仓库-添加远程库

先有本地库，然后关联远程仓库，并推送。

1、新建一个空白库，将本地仓库与之关联，然后将本地仓库的内容推送到Github仓库。

2、要关联一个远程库，需要执行以下命令

```
git remote add origin git@github.com:hellobei/learnngit.git
```

3、第一次推送：关联后，使用以下命令第一次推送master分支的所有内容

```
git push -u origin master
```

4、推送最新修改：没有特殊情况，第一次之后的最新修改

```
git push origin master
```

注：origin是远程仓库的名字，master是远程仓库的某个分支名字。

0129 远程仓库-从远程仓库克隆

一般来说，如果从零开发，最好的方式是先建立远程仓库，然后从远程仓库克隆，这样便于多人开发。

1、在Github上新建一个远程仓库，并且Initialize with a README.md文件。

2、在本地克隆这个远程库。

```
git clone git@github.com:hellobei/gitskills.git
```

3、注意git@github.com:hellobei/gitskills.git是SSH协议地址，相应的还有Https格式地址。但是默认的是SSH，也建议SSH，除非公司、单位只支持http协议。

0129 分支管理

分支的作用：开发一个新功能时，如果只在一个分支上进行，当你不断提交新的修改时，就会影响别人的工作。现在好了，独立开辟一个分支，我自己随便修改，确定完全搞定了，再讲此修改合并到原先的分支上去。

0129 分支管理-创建于合并分支

建立分支，例如名字为dev

```
git branch dev
```

切换到此分支

```
git checkout dev
```

（现在明白为啥切换文件的时候需要加--了吧）

建立并切换到此分支

```
git checkout -b dev
```

显示当前所有分支

```
git branch
```

在当前分支上合并其他分支（默认是Fast-forward模式）

```
git merge dev
```

删除dev分支

```
git branch -d dev
```

0129 分支管理-解决冲突

1、当git无法自动合并分支时，说明两者在修改上有冲突。解决办法很粗暴：在当前分支上修改为自己想要的文件内容，然后add并commit即可。

2、查看分支合并的图示

```
git log --graph
```

3、其中1是冲突后解决方案，那么实际中如何避免冲突呢？

答案是：在当前分支（如master）建立新分支（如dev）后，就不要再对master做任何操作了。

0129 分支管理-分支管理策略

1、合并时默认的模式是fast-forward，这样的模式没有保存原分支信息。

2、强制禁止fast-forward模式可以保留分支信息。保留的方式是增加一个commit，所以命令如下。

```
git merge <name> --no-ff -m "what note"
```

注意-m是对commit的标注

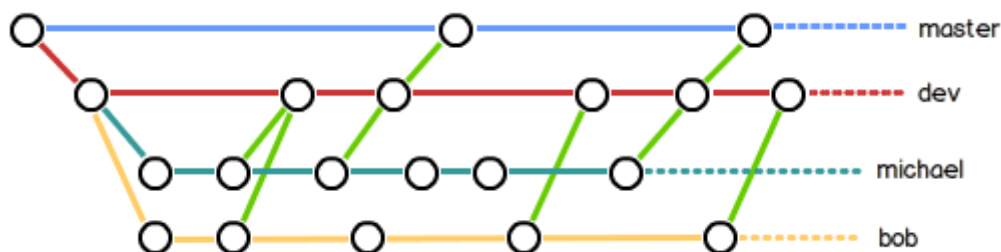
以下两幅图来个对比，使用ff模式：

```
E:\Git\learnGit>git log --pretty=oneline --graph
* ebba1bbfb52bbeafa2149e88e2c47121ab34e38 (HEAD -> master, newdev) test again on newdev
* 61d83873e2248a2eb2d7172b1638e977776eac my branch test
```

强行禁止ff模式：

```
E:\Git\learnGit>git log --pretty=oneline --graph
* 7814cfdcf6b88f6fcd5d49d10dffbe1fa3987afe (HEAD -> master) merge with --no-ff
* ebba1bbfb52bbeafa2149e88e2c47121ab34e38 (newdev) test again on newdev
* 61d83873e2248a2eb2d7172b1638e977776eac my branch test
```

3、分支管理策略：



0129 Bug分支与现场保护

1、修复bug时，新建一个bug分支，修改完毕后合并分支。

2、现场保护。当前工作区的修改没有完成，不能提交。但是又需要跳转到其它分支搞事情，那么当前任务怎么办？任其消失？不，我们将它现场保护起来：

```
git stash
```

当重新回来时，可以用恢复现场：

```
git stash pop
```

(相当于 git stash apply 与 git stash drop 两个命令的组合)

0129 Feature分支

1、新增一个功能，最好新建一个分支。

2、如果要丢弃一个还没有合并的分支，需要强行删除。强行删除是git帮助你防止误删。

```
git branch -D <name>
```

0129 多人协作

假设两台电脑都有了这个库learngit，现在需要共同在一个新的newdev分支上修改东西。

第一台电脑：将我的当前分支dev提交到github，使用命令

```
git push origin newdev
```

第二台电脑：

```
git checkout -b newdev //新建一个，最好和远程的newdev同名  
git pull origin newdev
```

多人协作的工作模式：

1、首先，试图提交自己的修改

```
git push origin newdev
```

2、发现冲突了，说明自上次pull下来修改至今的期间，有人push上去了自己的修改。那么我只能pull下来，来解决冲突。

```
git pull origin newdev
```

3、解决冲突，就像之前分支冲突一样，简单粗暴。

4、解决完之后，即可再次尝试1的步骤。

如果建立了本地分支和远程分支的连接关系，可以简化命令，还可以防止错误操作。

```
git branch --set-upstream <branch> origin/<branch>
```

建立这样的关系后，第三步的命令可以简化为

```
git pull
```

0129 标签管理

标签就是将commit复杂的编号标记为简单易记的版本号。

0129 标签管理-创建标签

1、新建一个标签，默认建立在当前HEAD版本

```
git tag <tagname>
```

2、新建一个标签，可以指定在哪个commit号上

```
git tag <tagname> commit_id
```

3、新建一个标签，可以指定标签信息

```
git tag -a <tagname> -m "what note"
```

4、新建一个标签，可以用PGP指定标签信息

```
git tag -s <tagname> -m "what note"
```


5、查看所有标签

```
git tag
```

6、查看某个标签的具体信息

```
git show <tagname>
```

0129 标签管理-操作标签

1、推送某个标签

```
git push origin <tagname>
```

2、推送所有标签

```
git push origin --tags
```

3、删除标签

```
git tag -d <tagname>
```

4、远程删除一个标签，先执行3，然后

```
git push origin :refs/tags/<tagname>
```

0129 使用GitHub

1、在GitHub上面可以任意克隆其他开源库，但是克隆到本地之后，可以在本地修改，不能给人家推送。（单向，只有读的权限）

2、在GitHub上面可以任意Fork其他开源库，到自己的账号上。然后克隆到本地，这样的仓库可以推送。（双向，读写权限）

3、可以推送pull request给源头官方仓库贡献代码。

0129 使用码云Gitee

1、Git默认远程库为origin，但是现在有了GitHub和码云两个远程库，所以可以修改远程库的名字。

2、允许私有库。

3、速度快。

0129 自定义Git

命令窗口颜色配置、忽略特殊文件、配置别名、搭建Git服务器等。

0129 自定义Git-忽略特殊文件

1、对于某些不想加入Git版本记录的文件，可以建立.gitignore文件（windows下不好建，可以在notepad++里新建然后另存为），在里面将特殊文件的格式排除掉。

2、可以将.gitignore文件加入版本控制。

0129 自定义Git-配置别名

1、自定义别名，加入xx是别名，xxxx是原名，则

```
git config --global alias.xx xxxx
```

2、--global是全局设置，即当前用户的所有仓库。不加的话，只是当前仓库。


3、配置文件。每个仓库的配置文件都在.git/config中。用户的global配置文件在用户文件夹中的.gitconfig文件中。

0129 自定义Git-搭建Git服务器

暂时用不到，详见廖雪峰网站。

0129 期末总结

Git cheat sheet, 常见命令与操作习惯养成。

 git-cheatsheet-中文注释.pdf
2018/1/30 12:27, 260 KB