一、教程:

https://www.liaoxuefeng.com/wiki/0013739516305929606dd18361248578c67b8067c8c017b000

二、环境问题:

ssh: connect to host github.com port 22: Connection timed out"

解决方法:

1. 生成秘钥 ssh-keygen -t rsa -C “[youremail@example.com](mailto:youremail@example.com)”

2. 找到git的安装目录，找到/etc/ssh/ssh\_config文件, 把如下内容复制到ssh\_config文件的末尾处

Host github.com

User YourEmail@qq.com

Hostname ssh.github.com

PreferredAuthentications publickey

IdentityFile ~/.ssh/id\_rsa

Port 443

其中User为登录github的账号名称。   
再次执行”ssh -T git@github.com”时，会出现提示如下，回车”yes”即可

3. 前往 GitHub 网站的"account settings" 依次点击"Setting -> SSH Keys"->"New SSH key"

Title处填写“id\_rsa.pub”或其他任意信息。 key处原样拷贝下面命令的打印 `~/.ssh/id\_rsa.pub` 文件的内容：cat ~/.ssh/id\_rsa.pub

4. 再次验证ssh -T [git@github.com](mailto:git@github.com)

Hi ykclin! You've successfully authenticated, but GitHub does not provide shell access.

出现这样的字样即表示链接成功

三、笔记：

1. 初始化一个Git仓库，使用git init命令。

2. 关联远程库：

首先在github网页上Create a new repository

在本地的learngit仓库下运行命令：关联到远程库

$ git remote add origin git@github.com:ykclin/learngit.git

3. 把本地库的所有内容推送到远程库上（第一次使用-u参数，后面push就不需要使用了）：

$ git push -u origin master

从现在起，只要本地作了提交，就可以通过命令：

$ git push origin master

4. 从远程库克隆

$ git clone git@github.com:ykclin/gitskills.git

场景： 对于某个git控制下的文件进行了修改，但是改的不满意，想退回到改之前的版本。假定该文件为 src/main/main.c

解决方法：

第一步： 在命令行中输入 git log src/main/main.c 得到该文件的commit 历史。

第二步： 复制需要回退版本的hash，在此假设我们回退到 d98a0f565804ba639ba46d6e4295d4f787ff2949 ,则复制该序列即可

第三步：checkout 对应版本。格式为 git checkout <hash> <filename>, 在此即为命令行中输入 git checkout d98a0f565804ba639ba46d6e4295d4f787ff2949 src/main/main.c

第四步： commit checkout下来的版本。 如： git commit -m "revert to previous version"

5. 删除一个文件

$ rm test.txt

$ git rm test.txt

$ git **commit** -m "remove test.txt"

6. 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>

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>

7. 查看log历史

$ git log

$ git reflog

$ git log --graph --pretty=oneline --abbrev-commit

8. stash功能，可以把当前工作现场“储藏”起来，等以后恢复现场后继续工作：

$ git stash

用git stash list命令看看：

$ git stash list

stash@{0}: WIP on dev: 6224937 add merge

一是用git stash apply恢复，但是恢复后，stash内容并不删除，你需要用git stash drop来删除；

另一种方式是用git stash pop，恢复的同时把stash内容也删了：

$ git stash pop

9. 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 <name>强行删除。

10. **多人协作的工作模式**通常是这样：

1. 首先，可以试图用git push origin branch-name推送自己的修改；
2. 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；
3. 如果合并有冲突，则解决冲突，并在本地提交；
4. 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。

这就是多人协作的工作模式，一旦熟悉了，就非常简单。

11. 远程库操作小结

* 查看远程库信息，使用git remote -v；
* 本地新建的分支如果不推送到远程，对其他人就是不可见的；
* 从本地推送分支，使用git push origin branch-name，如果推送失败，先用git pull抓取远程的新提交；
* 在本地创建和远程分支对应的分支，使用git checkout -b branch-name origin/branch-name，本地和远程分支的名称最好一致；
* 建立本地分支和远程分支的关联，使用git branch --set-upstream branch-name origin/branch-name；
* 从远程抓取分支，使用git pull，如果有冲突，要先处理冲突。

12． 增加文件夹

* 总结：
* **·**git add -A  提交所有变化
* **·**  git add -u  提交被修改(modified)和被删除(deleted)文件，不包括新文件(new)
* **·**  git add .  提交新文件(new)和被修改(modified)文件，不包括被删除(deleted)文件

13.标签

* 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。
* 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。

-------------------------------------------------------------------------------------------------------------

----------------------------------20231125--------------------------------------------------------------

----------------------------------------------------------------------------------------------------------------------

Linux 生成ssh key

ssh-keygen –t rsa (ssh-keygen 没有空格)

git clone <https://github.com/ykclin/learngit>

git@github.com:ykclin/learngit.git

git clone ssh://git@20.105.23.166/ykclin/learngit.git(ping github.com)

在一个分支testing\_branch(git checkout testing\_branch)

修改文件并push:

1. vi test.txt， 修改内容
2. git add test.txt
3. git commit test.txt –m ‘this is comment message’
4. git push origin testing\_branch

修改文件 git add 后反悔再rm

1. vi test.txt 修改内容
2. git add test.txt (不想add了 要重新修改)
3. git rm test.txt

误删除本地文件后恢复:

1. git rm test.txt
2. git reflog
3. git checkout HEAD@{0} test.txt