# 远程仓库GitHub

一、远程仓库

1、添加远程库

（1）在GitHub创建一个git仓库，create a new repo,填入Repository name

（2）创建一个本地仓库，与GitHub上的仓库关联

git remote add origin https://github.com/Wangdf0702/git-learn.git

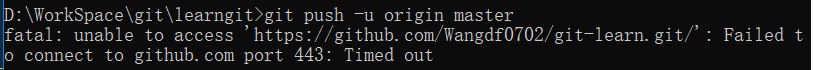
（3）推送本地库所有内容到远程库上

git push -u origin master

由于远程库是空的，第一次推送master分支时，加上了-u参数，Git不但会把本地的master分支内容推送的远程新的master分支，还会把本地的master分支和远程的master分支关联起来，在以后的推送或者拉取时就可以简化命令。

报错：fatal: unable to access 'https://github.com/Wangdf0702/git-learn.git/': Failed to connect to github.com port 443: Timed out

无法推送。



两个原因：**①不该加代理却不小心被代理软件加了代理**

**通过: git config --global http.proxy 查询到当前没有设置代理**

**(如果设置了代理，通过：git config --global --unset http.proxy,取消代理，然后再push)**

**②既然没有设置代理，那就去增加代理**

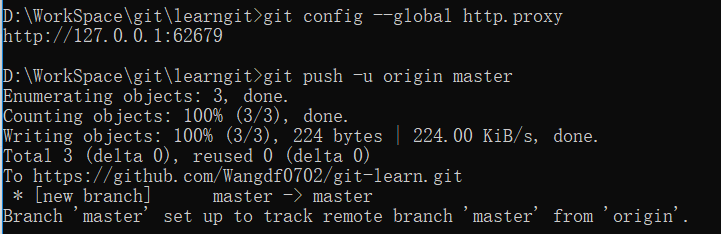
**第一步：IE ------Internet选项-------连接--------局域网设置，是否设置了代理服务器？，如果没有，那就没设置代理服务器。有 ，查看代理服务器地址和端口**

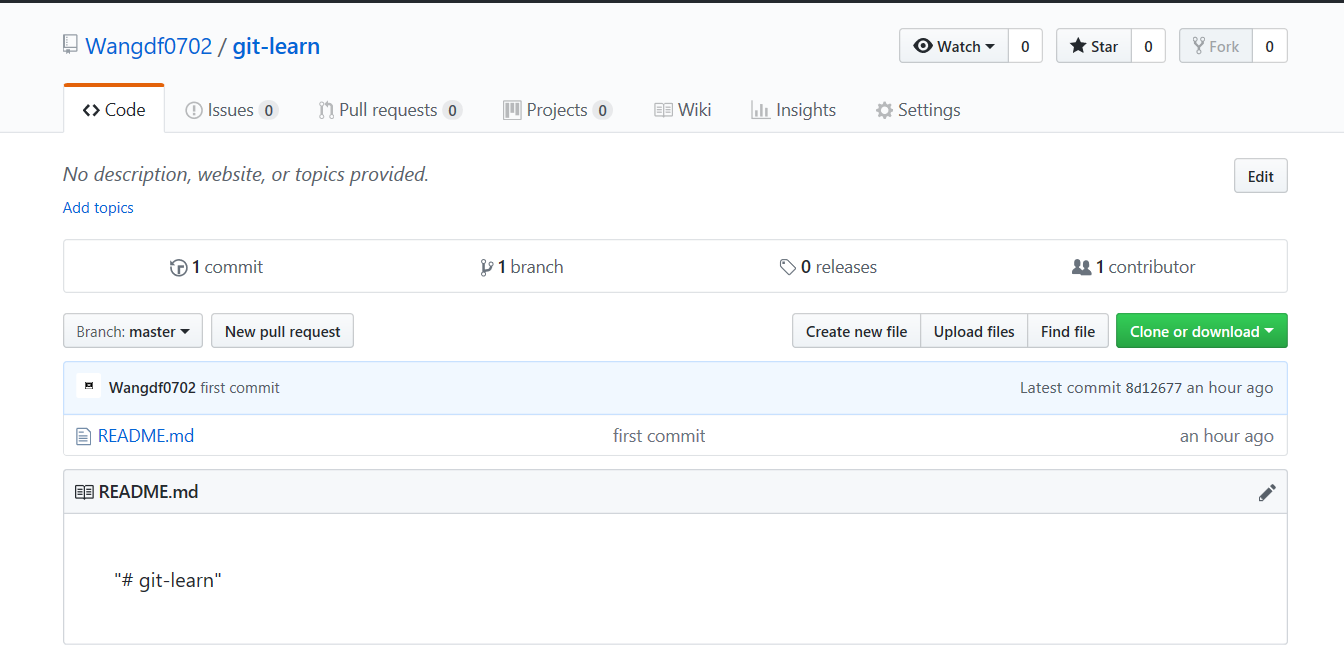
**第二步：去主用户目录下user------dell-------.gitconfig,打开添加**

**[http]**

**这里有一个TAB键proxy = http://127.0.0.1:62679**

**成功推送！此时就可以在远程库上看到 与本地一摸一样的内容**

****

****

以后通过： git push origin master命令就可以把本地master分支的最新修改推送至GitHub

（4）修改URL模式

Git支持多种协议，默认的git://使用ssh，但也可以使用https等其他协议。

使用https除了速度慢以外，每次推送都必须输入口令。

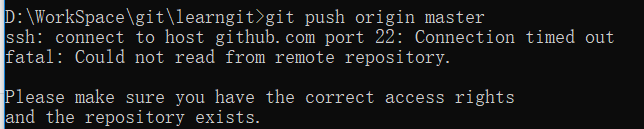
https->SSH(git原生协议)

**git remote -v：查看协议类型**

**git remote set-url origin git@github.com:Wangdf0702/git-learn.gi**t

验证URL是否改变git remote -v

实际操作：修改到ssh成功后，改变本地learngit仓库的README.ad,重新保存提交，push不成功



第一种原因：GitHub help（https://help.github.com/articles/changing-a-remote-s-url/）中给出解决办法是The next time you git fetch, git pull, or git push to the remote repository, you'll be asked for your GitHub username and password.

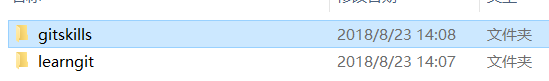
实际操作还是没有push到GitHub。

第二种原因：可能是防火墙的修改，就像你在一个网络下面一样。（在这种情况下，他们可能故意阻塞一些端口）。如果是这样的话，使用http协议而不是ssh。

push成功。

2、从远程仓库克隆到本地

git clone <http://github.com/Wangdf0702/gitskills.git>



二、分支管理

查看所有分支：git branch

1、分支管理策略

合并dev分支：git merge --no-ff -m "merge with no-ff" dev

--no-ff参数，表示禁用Fast forward，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。

因为本次合并要创建一个新的commit，所以加上-m参数，把commit描述写进去。

2、bug分支

每个bug都可通过一个新的临时分支来修复

(1)git stash:先将没有完成提交的工作，工作现场储藏起来

(2)git status:查看工作区是干净的

(3)确定在哪个分支上修复bug,就从哪个分支上创建临时分支

git checkout master(假设在需要在master上修复)

git checkout -b issue-101(创建并使HEAD指针指向issue-101分支)

(4)修复完成后保存提交，切换到master分支，完成合并，最后删除issue-101分支

git checkout master

git merge --no-ff -m "merge bug fix 101" issue-101

git branch -d issue-101

(5)回到之前的工作现场

1）git stash list:查看之前的工作现场

2)①git stash apply:恢复现场，但stash内容并不删除，删除stash内容用：git stash drop

②git stash pop:恢复的同时把stash内容也删了

3）git stash apply stash@{0}:可以多次stash，恢复工作现场时，现用git stash list查看，然后恢复指定的stash

3、Feature分支

开发一个新feature，最好新建一个分支

git branch -D new-branch-name

4、多人协作

两个人A、B同时开发

(1)A要在dev分支上开发，必须创建远程origin的dev分支到本地

git checkout -b dev origin/dev

(2)A修改完向origin/dev分支提交推送，此时B对同样的文件做了修改：cat .txt文件，并试图推送

git push origin dev

推送失败，因为B试图推送的提交与A的最新提交有冲突，

解决办法：git pull把最新的提交从origin/dev抓下来，在本地合并，解决冲突，再推送

1）git branch --set-upstream-to=origin/dev dev:先设定远程origin/dev分支与本地dev分支的关联

2）git pull：把最新的提交从origin/dev抓下来，在本地合并

3）手动解决冲突

4）提交推送

5、Rebase

(1)rebase操作可以把本地未push的分叉提交历史整理成直线；

(2)rebase的目的是使得我们在查看历史提交的变化时更容易，因为分叉的提交需要三方对比。

三、标签管理

发布一个版本时，通常先在版本库中打一个标签（tag），这样，就唯一确定了打标签时刻的版本。

标签是版本库的一个快照，指向某个commit指针，标签类似于分支，可以删除但不能移动。

1、创建标签

（1）切换到需要打标签的分支上

（2）git tag v1.0:在当前分支打下一个新标签（最新提交的commit）

（3）git tag:查看所有标签

（4）给以前提交的分支打标签：

git log --pretty=online --abbrev-commit

找到历史提交的commit id，给这次提交打标签：

git tag v0.5 commit\_id

（5）标签不是按时间顺序列出，按字母排序

git show tag\_name:查看标签信息

（6）创建带有说明的标签：

git tag -a v0.5 -m "blablabla..." commit\_id

用-a指定标签名，-m指定说明文字

注意：标签总是和某个commit挂钩。如果这个commit既出现在master分支，又出现在dev分支，那么在这两个分支上都可以看到这个标签。

2、操作标签

创建的标签都只存储在本地，不会自动推送到远程，所以打错标签在本地就可安全删除

（1）git tag -d v 0.5:删除本地标签

（2）git push origin tag\_name:推送标签至远程

（3）推送全部标签至远程：git push origin --tags

（4）删除远程标签：

先删除本地的,再远程删除：

git push origin :refs/tags/v0.5

四、自定义git

1、.gitignore的内容就是被忽略的文件，填进去需要忽略的文件名

（1）如果添加一个文件到Git添加不了，原因是这个文件被.gitignore忽略了：

git add -f a.class：强制添加a.class文件到git

（2）找出.gitignore可能写错的规则：

git check-ignore -v

（3）.gitignore文件本身要放到版本库里，可以对它做版本管理

2、配置别名

（1）git config --global alias.st status:以后st就表示status

checkout——co、commit——ci、branch——br

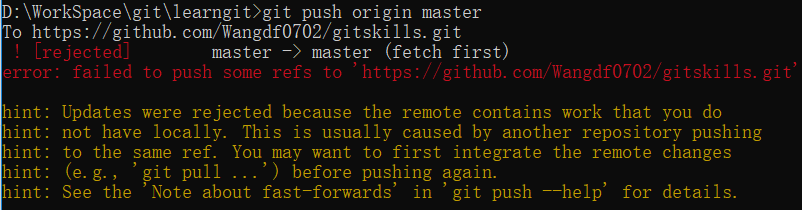
（2）配置一个git last,让其显示最近一次的提交：

git config --global alias.last 'log -1'

（3）git config --global alias.lg "log --color --graph --pretty=format:'%Cred%h%Creset -%C(yellow)%d%Creset %s %Cgreen(%cr) %C(bold blue)<%an>%Creset' --abbrev-commit"

实际操作中：

推送本地仓库learngit至远程仓库git-learn.git中，推送失败：



**1、从gitskills.git远程仓库切换到另一个git-learn.git远程仓库：**

**git remote set-url origin https://github.com/Wangdf0702/git-learn.git**

**2、学习笔记push成功后，再修改README.md文件，提交推送失败**

**fatal: unable to access 'https://github.com/Wangdf0702/git-learn.git/': OpenSSL SSL\_connect: SSL\_ERROR\_SYSCALL in connection to github.com:443**

**网上搜了一下，有可能是http代理问题，push的时候应该走ssh隧道的，但是设置的是http代理**

**解决办法：（1）删除设置的http代理：**

**git config --global --unset http.proxy**

**再推送，还是不成功**

**（2）因为之前已经知道本地防火墙不允许设置ssh协议，所以又重新设置了http代理：**

**.gitconfig文件里添加[http]**

这里有一个TAB键proxy = http://127.0.0.1:62679

再推送，成功！