GIT

git教程学习

**Git学习网站：：**

<https://backlog.com/git-tutorial/cn/contents/>

CSDN：很好的入门网站

https://blog.csdn.net/qq\_35246620/article/details/66973794

跟踪远程分支：

git branch --set-upstream-to=origin/远程分支名字

删除远程分支：git push origin --delete dev

Git rm后恢复文件：先Git reset，再Git checkout

创建一个空目录：mkdir learngit

初始化git：git init

查看一个隐藏.git目录：ls -ah

设置自己的邮箱和名字：git config --global user.email "yolo@ying31.com"

git config --global user.name "yolo"

编辑一个文件放在learngit目录或者子目录下：文件名为file1

把文件添加 到仓库（repository）：git add file1

把文件提交到仓库：git commit -m "wrote a file"

一次可以commit 多个文件，可以多次add不同的文件

修改file1文件

掌握仓库当前的状态：git status

查看修改的内容：git diff

cmd中换行符是^,而不是\

版本回退：

查看每个版本更改内容：git log或者git log --pretty=oneline，每次提交后都有一个提交号

当前版本是HEAD指针指着，上个版本HEAD^或者HEAD~1，上上个版本HEADD^^或者HEAD~2

往上100个版本，则是HEAD~100

完整命令为：git reset --hard HEAD^

回退到具体版本：git reset --hard 版本号

若不记得版本号，使用git reflog用来记录你的每一次命令

工作区和暂存区：

working directory：例如我们的learngit

暂存区：工作区有一个.git隐藏目录，这个是git的版本库

git版本库里有stage（暂存区），还有git为我们创建的第一个分支master，以及指向master的指针head

git管理的是每一次做的修改，而不是管理的文件

每次修改都需要先git add到暂存区，才能git commit 到仓库

撤销修改：git checkout -- readme.txt的意思是，把此文件在 工作区的修改全部撤销，两种情况：

一种是此文件的修改还在工作区，现在撤销修改就是和版本库一模一样的状态

一种是此文件已经添加到暂存区，又作了修改，现在撤销修改就回到添加到暂存区后的状态，，不会产生modified文件

修改添加到了暂存区，还没有提交，git同样可以：用命令git reset HEAD <file>可以把暂存区的修改撤销掉

重新放回工作区

删除文件：没有被添加到版本库就被删除的文件，是无法修复的！！！

在文件管理器中删除了文件，工作区和版本库不一致，有两个选择：

第一种，从版本库中删除文件，命令：git rm删掉，并且git commit

第二种情况：删除错了，版本库中还有，可以git checkout -- <file>，把误删的文件恢复到最新版本

git checkout其实使用版本库里的版本替换工作区中的版本，无论是删除还是修改都可以还原

远程仓库：git与github通过ssh方式连接

可以克隆出新的仓库，或者已有的本地仓库与远程仓库关联，然后把本地仓库内容推送到github仓库

关联远程仓库命令：git remote add origin git@github.com:yolotester/learngit.git

本地内容推送到远程命令：git push -u origin master

从现在起，只要本地做了提交，就可以通过命令：git push origin master

从远程库克隆：git clone git@github.com:yolotester/gitskills.git

进入gitskills目录看看，已经有远程库文件

分支管理：创建与合并分支：

创建并切换分支：git switch -c dev

查看当前分支：git branch

在dev分支上正常提交，比如对readme.md做修改。

dev分支工作完成，切换回master分支命令：git checkout master/git switch master

把dev分支的工作成果合并到master分支：git merge dev

合并完成后，删除分支：git branch -d dev

解决冲突：合并时出现冲突，需要先解决冲突，再提交

分支管理策略：

fast forward模式，合并后会丢掉分支信息

强制禁用fast forward模式，使用--no-ff参数，能够看出来曾经做过合并

实际开发中分支管理的几个原则：

1、master分支应该非常稳定，用来发布新版本

2、干活在dev分支上，某个时候，发布1.0版本，再把dev分支合并到master上，在master分支上发布1.0版本

bug分支：Git提供了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 status查看工作区，工作区是干净的，此时创建分支修复bug

首先确定在哪个分支上修复bug，假设在master上，从master创建临时分支

命令如下：git checkout master git checkout -b issue-101

查看工作现场：git stash list

恢复工作现场：第一种：git stash apply恢复，恢复后stash内容并不删除，需要用git stash drop来删除

第二种：git stash pop。恢复的同时，把stash内容删除

获此stash后，恢复的时候，恢复指定的stash命令：git stash apply stash@[0]

Git 提供了一个cherry-pick命令，能够复制一个特定的提交到当前分支

Feature分支：添加新功能时，为了不把主分支打乱，需要新建一个feature分支

在feature上面开发，完成后，合并分支，最后，删除该分支

命令：git switch -c feature-vulcan

开发完毕后：git add vulcan.py git commit -m 'add feature vulcan'

切回dev：git switch dev，顺利的话，合并，删除vulcan分支

若上级命令，新功能取消，则要销毁分支：git branch -d feature-vulcan

强行删除：git branch -D feature-vulcan

! [rejected] master -> master (fetch first)

error: failed to push some refs to 'git@github.com:yolotester/learngit.git'

意思是：本地分支，与远程分支不一致，需要先fetch

FERCH\_HEAD :是一个版本链接，指向着已经从远程仓库取下来的分支的末端版本

commit\_id : git commit操作后，会产生一个commit\_id，这是唯一标识一个版本的序列号。

git push后，这个序列号还会同步到远程仓库

git fetch取:这将更新git remote 中所有的远程仓库所包含分支的最新commit-id,

将其记录到.git/FETCH\_HEAD文件中

git fetch origin master:tmp

//在本地新建一个temp分支，并将远程origin仓库的master分支代码下载到本地temp分支

git diff tmp

//来比较本地代码与刚刚从远程下载下来的代码的区别

git merge tmp

//合并tmp分支到本地的master分支

git branch -d tmp

//如果不想保留tmp分支 可以用这步删除

git pull 拉<远程主机名> <远程分支名>:<本地分支名>

//取回远程主机某个分支的更新，再与本地的指定分支合并。

git-pull(1) =========== NAME ---- git-pull - Fetch from and integrate with another repository or a local branch SYNOPSIS

推送本地分支到远程分支的两种方式：



Git fecth和git pull的区别：



# **git push错误failed to push some refs to的解决：**

<https://blog.csdn.net/MBuger/article/details/70197532>

# **git解决non-fast-forward冲突：**

<https://blog.csdn.net/programerxiaoer/article/details/78585301>