位置：D:\\git

[1 创建版本库](#3554-1525746172792)

[1.1 创建空目录](#5994-1525746331211)

[1.2 将目录变成git库](#3024-1525746308169)

[2 添加文件](#4453-1525746510433)

[2.1 添加文件](#7148-1525746739518)

[2.2 丢弃修改和丢弃文件](#9198-1531719643461)

[3 修改文件](#3079-1525747435957)

[3.1 查看文件是否修改](#3489-1525747801607)

[3.2 版本回退](#1655-1525748612339)

[3.3 撤销修改](#2142-1525832191687)

[3.4 删除文件](#3087-1525832457224)

[4 远程仓库](#1532-1525834347228)

[4.1 设置github SSH Key](#4097-1525834626503)

[4.2 连接github远程仓库](#4472-1525922629484)

[4.3 连接github远程仓库](#4176-1525942597818)

[5 分支](#6439-1525943252725)

[5.1 创建与合并分支](#3917-1525943277598)

[5.2 解决冲突](#2558-1525945178154)

[5.3 分支合并--no-ff模式](#1720-1525943258748)

[5.4 git stash](#7978-1526005430085)

[5.5 远程分支](#3410-1525834354122)

[5.5.1 创建分支](#9145-1526020490961)

[5.5.2 抓取分支](#3945-1526020499993)

[5.5.3 删除分支](#6056-1530254415781)

[6 标签](#4661-1526023167429)

[6.1 添加标签](#2398-1526023173045)

[6.2 操作标签](#8480-1526024022724)

[7 参与github项目](#7936-1526025552305)

[8 自定义git](#2810-1526026685111)

[8.1 设置别名](#8891-1526026691311)

[9 新浪sae git 记住提交密码](#1766-1530177125179)

[1 使用如下命令,来储存您的密码凭据](#9095-1530177120288)

[2 配置用户邮箱和名字, 将引号内的内容替换（不需要引号）](#1520-1530177120288)

[错误解决](#8823-1525748448579)

[n.1 本地项目加入sourcetree中报错没有加入ssh](#2131-1530177135900)

[n.2 代码库文件覆盖本地文件(xshell)](#5610-1530253440955)

[n.3 清空多余文件（xshell）](#4476-1530253993085)

**1 创建版本库**

**1.1 创建空目录**

mkdir learngit ---创建目录

cd learngit ---进入目录

pwd ---显示文档当前目录

**1.2 将目录变成git库**

git init ---初始化git仓库，在当前目录生成隐藏的.git文件夹

**2 添加文件**

**2.1 添加文件**

在git仓库learngit或其子目录下添加readme.txt文件

git add readme.txt---将文件添加到git**暂存区**，即.git里。不存在的文件名，报错did not match any files

git add . ---添加当前目录所有文件

git commit -m "this is readme.txt" ---文件提交到git**仓库**。git commit是提交命令，-m后面是提交说明，说明尽可能清楚明了，以便查看。

git add file1.txt;git add file2.txt;git commit -m "commit two files" ---可以add多个文件，然后一次性commit。

**2.2 丢弃修改和丢弃文件**

<https://blog.csdn.net/leedaning/article/details/51304690>

git checkout ---丢弃修改

git checkout . ---丢弃修改的所有文件

git clean -df ---丢弃所有文件和文件夹

**3 修改文件**

修改readme.txt文件

**3.1 查看文件是否修改**

git status ---检查仓库当前的状态, modified: readme.txt说明readme.txt已经被修改。提交之后，nothing to commit, working tree clean说明没有被修改的文件，而且工作区是干净的。

git diff readme.txt ---查看被修改文件的修改部分。-是被修改的部分，+是修改后的部分。

**3.2 版本回退**

git log ---查看提交到仓库的版本，倒序排列。HEAD->master,当前版本和分支。

git log --pretty=oneline ---同上，只是这个命令版本只显示单行，易于观看。

git reset --hard HEAD^ ---返回上一个版本，HEAD^^上上版本；HEAD~100，往上第100个版本。HEAD是版本的定位。

cat readme.txt ---查看文件内容。

git reset --hard 0972312 ---返回指定版本,git reset --hard<commit id(版本号)>，0972312是commit id前几位，git可以自动查询。版本号可以由log来查询。

git reflog ---记录每一次提交命令以及该次提交的commit id 的前七位。由此可以将文件指定到任意版本。

英文状态下按q键 ---退出log

**3.3 撤销修改**

git checkout -- readme.txt ---撤销工作区的修改，即用仓库的版本代替工作区版本。执行此命令之前status查看：Changes not staged for commit。执行此命令之后status查看：nothing to commit, working tree clean

git checkout . ---撤销所有修改

git reset HEAD readme.txt ---撤销暂存区的修改。执行此命令之前status查看：Changes to be committed，执行此命令之后status查看：Changes not staged for commit。再执行一次上条命令，修改完成。

**3.4 删除文件**

在learngit目录里面加入test.txt文件，并add，commint到git仓库

rm test.txt ---删除工作区文件，和在文件夹中删除文件相同。status查看： deleted:test.txt。如果工作区误删了文件，可以用git checkout -- readme.txt恢复。

git rm test.txt ---删除git仓库的文件。

**4 远程仓库**

**4.1 设置github SSH Key**

gitbuh账号：1145728724@qq.com或者catsforeset，密码：ly\*\*\*38

（1）

ssh-keygen -t rsa -C "1145728724@qq.com" ---设置ssh key。不用设置key，一路回车。Enter file in which to save the key (/c/Users/Administrator/.ssh/id\_rsa)说明id\_rsa和id\_rsa.pub在目录c/Users/Administrator/.ssh中。

（2）

github->setting->SSH and GPG keys->new ssh key(创建ssh key)，随意设Ttitle，将id\_rsa.pub内容粘贴到Key中。

**4.2 连接github远程仓库**

首先要在github添加一个仓库learngit

（1）

git remote add origin git@github.com:catsforeset/learngit.git ---将现有的仓库加入github。前提是项目已经init，add并commit。

（2）

git push -u origin master ---将仓库推送到github中，git push推送，第一次推送master时，加上-u参数。Are you sure you want to continue connecting (yes/no)?yes(输入yes)。完成后刷新git仓库即可看到上传的文件。

git push origin master ---本地git仓库加入github仓库后，用此命令推送每次本地修改到github仓库。分布式版本库完成。github的push速度比较慢。

git push ---推送到当前分支的远程仓库里。

**4.3 连接github远程仓库**

创建一个github仓库gitskills，选择Initialize this repository with a README选项生成README文件

git clone git@github.com:catsforeset/gitskills.git ---克隆gitskills仓库到本地。成功后在项目中可看到gitskills和其README.md文件，或者 git clone https://github.com/catsForeset/gitskills.git ，https://github.com/catsForeset/gitskills.git是git仓库地址，在github的Clone and download选项中可以看到。

**5 分支**

**5.1 创建与合并分支**

git checkout -b dev ---创建新分支dev并切换到改分支。相当于git branch dev;git checkout dev两个命令的结合。

git branch --set-upstream-to=origin/luoyan\_dev luoyan\_dev ---分支合并到远程分支

git branch ---查看当前分支。

git branch -a ---查看所有分支。

git fetch ---更新分支。

git checkout master ---切换到master分支。

git merge dev ---将dev分支合并到master分支（首先切换到master分支）。

git branch -d dev ---删除dev分支。

**5.2 解决冲突**

创建一个新分支feature1，往readme.txt加入数据并commit提交。切换到master分支，同样往readme.txt加入数据并commit提交，执行git merge feature1出错。

git status查看，报错Your branch is ahead of 'origin/master' by 2 commits.

查看readme.txt文件可查看冲突部分。Git用<<<<<<<，=======，>>>>>>>标记出不同分支的内容。

<<<<<<<HEAD（当前分支）

冲突部分1

=======

冲突部分2

>>>>>>>feature1（冲突分支）

手动改冲突部分并去掉<<<<<<<，=======，>>>>>>>标记，并commit当前分支（master），冲突解决。

git log --graph --pretty=oneline --abbrev-commit ---查看分支情况。

**5.3 分支合并--no-ff模式**

git merge feature1-fast forword模式合并分支feature1,看不出来有合并。

\* 0c98210 git merge

\* 946246f git origin

\* dadb1b9 add test.txt

\* 0972312 add new line

\* 204fce2 change readme.txt

\* 03106fb this is readme.txt

git merge --no-ff -m "merge with --no-ff" ---普通模式合并分支，看得出来合并效果。

\* 70c536e merge with --no-ff

|\

| \* cb63bee merge --no-ff

|/

\* 0c98210 git merge

\* 946246f git origin

\* dadb1b9 add test.txt

\* 0972312 add new line

\* 204fce2 change readme.txt

\* 03106fb this is readme.txt

**5.4 git stash**

feature1进行到一半不能提交，但需要去解决其他问题。用stash来储存feature1的内容，创建issue-101分支解决好问题之后，再返回feature1分支读取stash内容，继续完成未完成的工作。

git stash ---暂存。

git stash list ---查看暂存列表。

git stash pop ---读取暂存内容并删除暂存。等于git stash apply;git stash drop的结合。

**5.5 远程分支**

**5.5.1 创建分支**

git remote ---查看远程仓库

git remote -v ---查看远程仓库详情。有fetch和push权限

git push origin master ---推送master分支到远程仓库，邮件是1145728724@qq.com，密码是ly\*\*\*38，github账号是catsforeset，密码是ly\*\*\*38。在将本地仓库连接到远程仓库时，本地master分支默认推送到远程master分支。

git push origin dev ---推送dev开发分支。配置同上。完成后git branch -a查看，出现remotes/origin/dev，远程分支创建成功。

**5.5.2 抓取分支**

git checkout -b dev origin/dev ---抓取远程分支。抓取到远程分支后，可在分支里commit和push工作内容。

git push origin dev ---将工作内容推送到远程分支。配置如5.5.1。

git pull ---从远程拉取分支内容。如果pull失败，说明没有指定本地分支和远程分支的链接，执行如下命令。

git branch --set-upstream dev origin/dev ---设置本地分支dev和远程分支origin/dev的链接。执行成功的话可以执行git pull命令。或者如下。

git branch --set-upstream-to=origin/716api orgin/716api

**5.5.3 删除分支**

git branch -r -d dev ---删除分支dev

git branch -r -d origin/dev ---删除远程分支dev

**6 标签**

**6.1 添加标签**

git tag v1.0 ---添加标签。首先切换到打标签的分支。默认标签打在最新的commit上。

git tag ---查看标签。tag按字母排序。

git tag v0.9 6084e53 ---对指定的某一commit加入标签6084e53是commit id的缩写。

git tag -a v0.1 -m "version 0.1 released" 03106fb ---创建带有说明的标签，用-a指定标签名，-m指定说明文字。

git show v0.1 ---查看tag详情。

git tag v0.2 -m "signed version 0.2 released" 204fce2 ---通过-s用私钥签名一个标签。

**6.2 操作标签**

git tag -d v0.1 ---删除标签v0.1。

git push origin v1.0 ---推送标签v1.0到远程仓库。

git push origin --tags ---推送所有标签到仓库。

git push origin :refs/tags/v0.9 ---先删除本地git tag -d v0.1，再执行本条命令，删除远程tag v0.9。

**7 参与github项目**

进入项目如bootstrap，<https://github.com/twbs/bootstrap>。点击Fork克隆项目到自己的github仓库，再用git clone的方式将自己github仓库的项目克隆到本地。在本地修改并推送代码。如果不fork别人的项目到自己的仓库就clone到本地仓库，则没有推送修改的权限。如果希望bootstrap的官方库能接受自己的修改，可以在GitHub上发起一个pull request。是否接受这个request由其官方决定。

**8 自定义git**

**8.1 设置别名**

git config --global alias.st status ---设置status别名为st。其他命令也可以这样设置别名。--global参数是全局参数，也就是这些命令在这台电脑的所有Git仓库下都有用。

git config --global alias.last 'log -1' ---设置显示最后一次提交信息。

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" ---显示提交

**9 新浪sae git 记住提交密码**

1 使用如下命令,来储存您的密码凭据

$git config --global credential.helper wincred

2 配置用户邮箱和名字, 将引号内的内容替换（不需要引号）

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

成功，下次就不用再输密码了

10 查看git的config文件（包括用户和邮箱等）

git config --list

11 忽略文件

如果文件没有被追踪过，则直接在.gitignore里面下入这个文件/文件夹；如果文件以及被追踪过，则：

（1）.gitignore设置这个文件，

（2）把此文件备份

（3）删除此文件并上传到远程，然后把备份文件加入到项目中。此时文件再修改也不会被git追踪。

参考：<https://blog.csdn.net/afanyusong/article/details/60581200>

**错误解决**

**n.1 本地项目加入sourcetree中报错没有加入ssh**

解决：sourcetree->工具->选项->ssh客户端，设置为OpenSSH。此时SSH秘钥自动选择4.1设置好并生成的id\_rsa文件（一定要将4.1配置好）。点击确定，错误解决。

**n.2 代码库文件覆盖本地文件(xshell)**

error: Your local changes to the following files would be overwritten by merge:
        protected/config/main.php
Please, commit your changes or stash them before you can merge.

解决：

代码库文件覆盖本地文件

git reset --hard

git pull

**n.3 清空多余文件（xshell）**

error: The following untracked working tree files would be overwritten by checkout:

解决：

git clean  -d  -fx ""

其中

x  -----删除忽略文件已经对git来说不识别的文件

d  -----删除未被添加到git的路径中的文件

f  -----强制运行

n.4 <https://blog.csdn.net/shiren1118/article/details/7761203>

n.5 fatal: refusing to merge unrelated histories

解决：

git pull origin master --allow-unrelated-histories

n.6 Please enter a commit message to explain why this merge is necessary;

解决：

1.按键盘字母 i 进入insert模式

2.修改最上面那行黄色合并信息,可以不修改

3.按键盘左上角"Esc"

4.输入":wq",注意是冒号+wq,按回车键即可