参考教程：廖雪峰的官方网站<https://www.liaoxuefeng.com/wiki/0013739516305929606dd18361248578c67b8067c8c017b000>

一、Windows安装Git

在Windows上使用Git，可以从Git官网直接[下载安装程序](https://git-scm.com/downloads)(https://git-scm.com/downloads)，然后按默认选项安装即可。

安装完成后，在开始菜单里找到“Git”->“Git Bash”，蹦出一个类似命令行窗口的东西，就说明Git安装成功！



安装完成后，还需要最后一步设置，在命令行输入：

$ git config --global user.name "Your Name"

$ git config --global user.email "email@example.com"

因为Git是分布式版本控制系统，所以，每个机器都必须自报家门：你的名字和Email地址。

注意git config命令的--global参数，用了这个参数，表示你这台机器上所有的Git仓库都会使用这个配置，当然也可以对某个仓库指定不同的用户名和Email地址。

二、创建版本库

什么是版本库呢？版本库又名仓库，英文名**repository**，你可以简单理解成一个目录，这个目录里面的所有文件都可以被Git管理起来，每个文件的修改、删除，Git都能跟踪，以便任何时刻都可以追踪历史，或者在将来某个时刻可以“还原”。

所以，创建一个版本库非常简单，首先，选择一个合适的地方，创建一个空目录：

$ mkdir learngit //在当前目录下创建文件夹learngit

$ cd learngit //切换到learngit目录下

$ pwd //显示当前目录

/Users/michael/learngit

第二步，通过git init命令把这个目录变成Git可以管理的仓库：

$ git init //将当前目录初始化成一个Git版本库

Initialized empty Git repository **in** /Users/michael/learngit/.git/

可以发现当前目录下多了一个.git的目录，这个目录是Git来跟踪管理版本库的。

注：创建版本库不一定非得是空白目录，可以将任意目录初始化成版本库。

三、提交文件到Git仓库

注：准备提交到Git仓库的文件一定要放在版本库（例：learngit）目录下（子目录也行），因为这是一个Git仓库，放到其他地方Git再厉害也找不到这个文件。

将版本库目录下的文件提交到git仓库需要两步：

第一步，用命令git add告诉Git，把文件添加到仓库：

$ git add readme.txt //将readme.txt文件添加到仓库（暂存区）

第二步，用命令git commit告诉Git，把文件提交到仓库：

$ git **commit** -m "wrote a readme file"

[master (root-**commit**) eaadf4e] wrote a readme file

1 file changed, 2 insertions(+)

**create** mode 100644 readme.txt

git commit命令，-m后面输入的是本次提交的说明，可以输入任意内容。

四、远程仓库

建立本地仓库与远程仓库（GitHub）之间的连接：

第1步：创建SSH Key。在用户主目录下，看看有没有.ssh目录，如果有，再看看这个目录下有没有id\_rsa和id\_rsa.pub这两个文件，如果已经有了，可直接跳到下一步。如果没有，打开Shell（Windows下打开Git Bash），创建SSH Key：

$ ssh-keygen -t rsa -C "youremail@example.com"

通过如上命令生成SSH Key，一路回车即可。

第2步：登陆GitHub，打开“Account settings”，“SSH Keys”页面：

然后，点“Add SSH Key”，填上任意Title，在Key文本框里粘贴id\_rsa.pub文件的内容：

第3步：在GitHub上创建与本地仓库名称相同的Git仓库

第4步：建立本地仓库与GitHub的连接

$ git remote add origin git@github.com:sunsir12138/learngit.git

注：sunsir12138为GitHub用户名，lerangit.git为仓库名称

第5步：将本地仓库内容推送至远程仓库

$ git push -u origin master

把本地库的内容推送到远程，用git push命令，实际上是把当前分支master推送到远程。

由于远程库是空的，我们第一次推送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。

五、分支管理策略

通常，合并分支时，如果可能，Git会用Fast forward模式，但这种模式下，删除分支后，会丢掉分支信息。

如果要强制禁用Fast forward模式，Git就会在merge时生成一个新的commit，这样，从分支历史上就可以看出分支信息。

$ git merge --no-ff -m "merge with no-ff" <name> //<name>被合并分支

注：请注意--no-ff参数，表示禁用Fast forward

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。

用git log看看分支历史

Git常用命令：

1、初始化一个Git仓库

$ git init //将当前目录初始化成一个Git版本库

2、将文件添加到Git仓库，分两步

$ git add <file> //将文件添加到仓库（暂存区）,<file>代表提交文件名，可反复多次使用，添加多个文件

$ git commit –m <message> //完成提交，<message>是对本次提交的描述，””。

3、查看仓库状态

$ git status

4、查看文件修改内容

$ git diff <file>

5、查看Git仓库提交日志

$ git log //显示从最近到最远的提交日志

$ git log --pretty=oneline //显示简略的提交日志，仅显示 commit\_id和提交描述

6、版本回退

$ git re**set** --hard HEAD^ //回退到上一版本，主要^号。

$ git re**set** --hard commit\_id //回退到指定版本，commit\_id只需要输入前几位

7、查看命令历史记录

$ git reflog //记录每一次命令

8、丢弃工作区的修改

$ git checkout -- <file> //撤销指定文件的本次的全部修改，让这个文件回到最近一次git commit或git add之后的状态

注：git checkout其实是用版本库里的版本替换工作区的版本，无论工作区是修改还是删除，都可以“一键还原”。

9、撤销暂存区的修改

$ git re**set** HEAD <file> //把暂存区的修改撤销掉（unstage），重新放回工作区。相当于回退git add 命令

9、删除版本库中的文件

$ git rm <file>

$ git **commit** -m <message>

10、生产SSH Key

$ ssh-keygen -t rsa -C "youremail@example.com"

11、建立本地仓库与GitHub的连接

$ git remote add origin git@github.com:sunsir12138/learngit.git

注：将本地仓库与远程仓库建立连接的前提是远程仓库已创建相同名称仓库（注：也可不相同？？），sunsir12138为GitHub用户名，lerangit.git为仓库名称

12、推送本地仓库内容至远程仓库

$ git push -u origin <name> //<name>要推送内容的当前分支名称

注：将当前master分支内容推送至远程master分支。首次推送命令需要加 –u，之后可以省略

13、克隆远程仓库中已有的Git仓库到本地

$ git clone [git@github.com:sunsir12138/gitskills.git](mailto:git@github.com:sunsir12138/gitskills.git)

14、创建分支

$ git branch <name> //<name>新分支名称，新分支是从当前所处分支中分支出去的

15、切换分支

$ git checkout <name>

注：

$ git checkout -b <name> //该命令相当于创建分支和切换分支的组合

16、查看分支

$ git branch

\* dev

master

注：\* 号代表当前所处分支

17、合并分支

$ git merge <name> //将分支名为<name>的分支合并到当前所处分支

18、删除分支

$ git branch -d <name>

19、查看分支合并情况

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

20、禁用fast-forward模式合并分支

$ git merge --no-ff -m "merge with no-ff" <name>

注：“merge with no-ff”合并分支描述，<name>被合并分支名称

21、隐藏当前分支已修改内容

$ git stash

注：将当前分支已修改的内容隐藏起来，通过git status命令查看当前分支处于未修改内容状态，常用与修复bug

22、查看所有隐藏

$ git stash list

stash@{0}: WIP on dev: f52c633 add merge

注：显示所有的隐藏

23、恢复隐藏内容

$ git stash pop

24、强行删除未合并过的分支

$ git branch –D <name> //强行删除未执行过合并操作的分支

25、查看远程库的信息

$ git remote -v

26、本地创建与远程分支对应的分支

$ git checkout -b <name> origin/<name>

注：在本地创建和远程分支对应的分支，本地和远程分支的名称最好一致

27、建立本地分支和远程分支的关联

$ git branch --set-upstream-to <name> origin/<name>

28、抓取远程分支最新提交

$ git pull

注：将当前分支对应的远程分支的最新提交抓取到本地，注意处理冲突

29、创建标签

$ git tag <tagname> //例：$ git tag v1.0

注：标签：标记版本，标签默认打在当前分支最新commit上

30、查看所有标签

$ git tag //标签不是按照创建时间排序，按照字母排序

31、给某个指定的commit创建标签

$ git tag <tagname> commit\_id //；例：$ git tag v0.9 f52c633

32、查看标签信息

$ git **show** <tagname>

33、创建带有说明的标签

$ git tag -a <tagname> -m "version 0.1 released" commit\_id

注：用-a指定标签名，-m指定说明文字

34、删除标签

$ git tag -d <tagname>

35、推送标签至远程仓库

$ git push origin <tagname>/--tags

注：推送标签至远程仓库可以推送指定的标签，也可以通过--tags一次性推送全部尚未推送的标签

36、删除远程标签

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