参考链接：<https://www.liaoxuefeng.com/wiki/896043488029600/896202815778784>

sudo apt-get install git

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

git config --global user.email [email@example.com](mailto:email@example.com)

git init //初始化git仓库

git add readme.txt //把文件添加到仓库

git commit -m "wrote a readme file" //文件提交到仓库，-m：添加记录说明

1 file changed, 2 insertions(+)：1个文件被改动，插入了两行内容

为什么Git添加文件需要add，commit一共两步呢？因为commit可以一次提交很多文件，所以你可以多次add不同的文件，比如：

git add file1.txt

git add file2.txt file3.txt

git commit -m "add 3 files."

//---------------

git status //命令输出告诉我们，工作区和暂存区哪些文件被修改过了

git diff readme.txt //看具体修改了什么内容，未提交的修改和仓库最新版

提交修改和提交新文件是一样的两步：git add和git commit

1 file changed, 1 insertion(+), 1 deletion(-)：一个文件删除

git log

git log --pretty=oneline //查看从最近到最远的提交日志列表

git reset --hard HEAD~1 //往上回退一个版本，回退后当前版本消失（未推送远程库时）

git reset --hard 1094a //回退到某个版本，1094a是版本号前几位

git reflog //查看历史操作，尤其查看丢失文件的版本号

创建的git根目录：工作区；.git目录：版本库；版本库里面的stage/index：暂存区

git checkout -- readme.txt //让这个文件回到最近一次时的状态（还未使用git commit或git add 时，撤销修改）

①未暂存，git checkout -- readme.txt

②暂存了，先git reset HEAD <文件>，再①

③回退版本（未推送至远程库）

先rm，再git status ，再git rm <file>或者恢复 //删除工作区文件

ssh-keygen -t rsa -C [1210267211@qq.com](mailto:1210267211@qq.com) //家目录下.ssh目录，生成id\_rsa是私钥，不能泄露出去，id\_rsa.pub是公钥，可以放心地告诉任何人。

git remote add origin git@github.com: qqngwzlb/learning\_Git.git（ssh）

git remote add origin https://github.com/qqngwzlb/learning\_Git.git

//链接到远程库，origin是给远程库起的名字，可以自定义

git push -u origin master

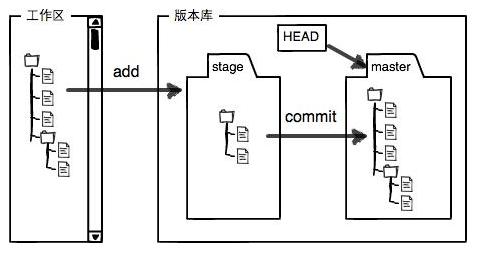
//我们第一次推送master分支时，加上了-u参数，Git不但会把本地的master分支内容推送的远程新的master分支，还会把本地的master分支和远程的master分支关联起来，在以后的推送或者拉取时就可以简化命令：git push origin master

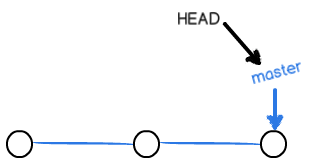
git remote -v //查看远程库信息

git remote rm origin //解除了本地和远程的绑定关系，。远程库本身并没有任何改动。要真正删除远程库，需要登录到GitHub，在后台页面找到删除按钮再删除

git clone https://github.com/qqngwzlb/gitskills.git //克隆gitskills库到本地工作区

分支管理原理见：上述链接->创建与合并分支





git checkout -b dev //表示创建并切换到dev分支

git checkout master //表示切换到master分支==git switch -c dev（最新版git），git switch master切换

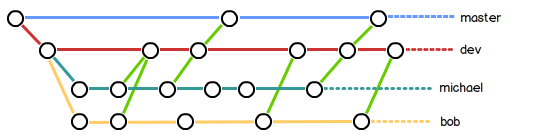
git branch //查看当前分支，-a可以看见远程分支

git merge dev //合并指定分支到当前分支

git branch -d dev //删除dev分支



当创建的新分支与master分支同时修改了同一个文件，但修改的内容不一致，各自提交后，是无法进行分支合并的，必须将内容修改为一致。



你和你的小伙伴们每个人都在dev分支上干活。

**git merge --no-ff -m "merge with no-ff" dev** //保留历史分支连接情况信息

**git log --graph --pretty=oneline --abbrev-commit** //查看分支连接情况

在某个分支正在修改文件时，要去修复该文件某些其他问题：

**git stash**保留现场 -》 修改问题 -》 暂存并提交 -》 切换到主分支并合并-》切换到该分支**git stash list**保留的现场列表、**git stash apply stash@{0}**恢复、**git stash drop stash@{0}**删除记录

**git cherry-pick 4c805e2** //复制修改到当前分支，可能和merge一样发生冲突，手动修改

feature-xx分支：每添加一个新功能，最好新建一个分支，然后合并

**git remote** //查看远程库的信息，名称

fetch：抓取、push：推送、Fork他的仓库：复制别人的仓库到你的GitHub帐号下

**git push origin master**  //推送某个分支到远程库

多人协同时，提交的文件冲突：先**git pull**抓取最新的提交，若失败再**git**

**branch --set-upstream-to dev origin/dev**建立（本地分支）与（远程分支）连接，再**git pull**，无法完成的话则需要手动解决文件冲突，commit后，再试着push

本地add，commit；远程push

rebase：变基操作，Git的提交历史不能是一条干净的直线

**git rebase** //整理提交历史线

打标签：**git log --pretty=oneline --abbrev-commit** //历史提交的commit id

**git tag v1.0 f52c633** //给f52c633希哈码的commit打标签V1.0，不加希哈码，默认标签是打在最新提交的commit上的，也就是HEAD目前所指

**git tag** //查看标签

**git show V1.0** //显示V1.0标签的文件信息

**git tag -a tagname -m "version info"** //用-a指定标签名，-m指定说明文字

**git tag -d v0.1** //删除标签（创建的标签都只存储在本地，不会自动推送到远程）

**git push origin v1.0**  //推送某个标签到远程

**git push origin --tags** //一次性推送全部尚未推送到远程的本地标签

删除远程标签，先删除本地标签，再**git push origin :refs/tags/tagname**

一般要使用别人的仓库，再github网页去Fork，在命令行clone到本地计算机！—-—

**git config --global color.ui true** //显示界面设置

.gitignore文件：在Git工作区的根目录下创建，然后把要忽略的文件名填进去，Git就会自动忽略这些文件，创建完成commit！

Gitee：国内的Git托管服务