**Linux下安装git命令：**

sudo apt-get install git

sudo apt-get install git-core 🡪 该命令是老版本的Linux的系统使用，一般执行上面一句就可以。

查看git用户名和邮箱名：

git config user.name

git config user.email

修改git用户名和邮箱名：

git config --global user.name “用户名”

git config --global user.email “用户邮箱”

解释：--global参数是指本地所有的仓库都配置成这个用户名和用户邮箱。

本地新建一个文件夹做git管理的仓库，然后在这个文件夹下执行git init，该文件夹便成了git仓库。

提交步骤：

步骤一：git add file1.txt file2.txt 🡪 一次可以add多个文件

步骤二：git commit -m “log message” 🡪双引号内是log信息，可以中文。

修改了文件内容，想看看与版本库中提交过的文件有什么变化；

执行：git diff file1.txt 🡪查看file1.txt变化的内容

查看log信息

git log 🡪 详细log信息，包括用户信息和提交时间等。

git log --pretty=oneline 🡪 单行信息，简约易看，没有用户信息和时间等。

**回退版本（时光穿梭）：**

git 中是以HEAD表示当前版本，上一个版本用HEAD^表示，上上个版本用HEAD^^,当然如果要回退100个版本，这样加100个^的做法是不科学的，可以写成HEAD~100，这样就表示100个版本，具体回退命令如：git reset --hard HEAD^

当你回退了版本之后，git status查看到最新的版本的log信息已经不在了，如果还想恢复到最新那个版本的话，往上找到对应的commit id号，执行回退命令加上对应的commit id号一样可以恢复到指定版本：git reset --hard 1094a，commit id号只需要前面几位可以准确定位版本就可以，可以不用写全。

但是如果往上翻不到commit id，要执行命令：git reflog，git会记录每一次命令，执行完就可以看到对应的commit id。

**撤销修改：**

git checkout -- filename.txt

该命令可以使工作区的修改内容复原，要么恢复到跟暂存区的内容一样（这种是之前git add了，但是还没git commit），要么恢复到版本库一样（这种是之前已经git commit过的）。

git reset HEAD filename.txt

该命令是可以让已经add到暂存区的内容回退到工作区，是以HEAD该版本回退的，执行完了，现在可以执行上面git checkout -- filename.txt把工作区的内容撤销掉修改。

**删除文件**

git rm 🡪 git commit –m “log信息”

git rm 与git add相似，当你工作区删掉了一个文件，然后git rm filename.txt是可以同步到暂存区，然后再git commit -m “删掉filename.txt”,这样就可以同步删掉版本库中的文件；另一种是不小心删掉了工作区的文件，想要恢复，这就可以版本库同步回本地工作区的方法，git checkout -- filename.txt

笔记：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”。

**远程仓库**

ssh-keygen -t rsa -C "994189964@qq.com"

在主目录下执行该命令，是创建SSH key用的，三下回车就可以了，把密钥添加到远程库上即可，比如：GitHub。详细的方法可以参考百度，细节不便啰嗦。

git remote add origin https://github.com/ZhifengPan/learngit.git

origin 是自己给远程库定义的名字，执行以上的命令就是把本地的仓库与远程库建立了联系。

git push –u origin master

这命令是指把本地master分支内容推送到远程库，初次推送加上-u，把本地master分支和远程库的master关联起来，后面推送就默认推到master，不用再加-u。

git push origin 本地分支名:远程分支名

将本地当前分支 推送到 远程指定分支上（注意：pull是远程在前本地在后，push相反）

git push origin 本地分支名

将本地当前分支 推送到 与本地当前分支同名的远程分支上（注意：pull是远程在前本地在后，push相反）

git push origin

将本地当前分支 推送到 与本地当前分支同名的远程分支上(需先关联远程分支)

git pull origin远程分支名:本地分支名

将远程指定分支拉取到本地指定分支上

git pull origin 远程分支名

将远程指定分支拉取到本地当前分支上。

git pull origin

将与本地当前分支同名的远程分支 拉取到本地当前分支上(需先关联远程分支)

git clone <https://github.com/ZhifengPan/gitskills.git>(这是远程库的地址)

克隆远程仓库，多人协作的时候，可以通过这种方式下载远程库工程文件

**分支管理**

git checkout –b dev （dev是分支名）

创建dev分支，并切换到dev的分支，创建和切换连续完成了。

git branch dev （dev是分支名）

创建分支dev

git checkout dev (dev是分支名)

切换分支dev

git branch

查看分支，带\*号表示当前分支

git merge dev

合并指定分支到当前分支

git branch –d dev (dev是要删除的分支名)

删除指定分支

Tips：因为创建、合并和删除分支非常快，所以Git鼓励你使用分支完成某个任务，合并后再删掉分支，这和直接在master分支上工作效果是一样的，但过程更安全。

git switch –c dev

创建并切换到dev分支

git switch master

切换到master分支

Note：switch是新版本git才支持的命令，旧版本不支持。

**解决冲突**

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

查看分支合并情况，带图示。

当在两个分支都对同一个文件修改，然后还要合并他们，就会有冲突，这时候切换分支会提示错误，我们要打开冲突的文件，修改冲突项，然后保存重新提交，这时候就已经把两个分支重新合并在一起。

**分支管理策略**

git merge --no-ff -m "【merge with no-ff】" dev

合并分支时，加上--no-ff参数就可以用普通模式合并，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。

git stash

保存工作区状态，再去做其他事

git stash list

查看stash列表

一是用git stash apply恢复，但是恢复后，stash内容并不删除，你需要用git stash drop来删除；另一种方式是用git stash pop，恢复的同时把stash内容也删了

可以通过list指定恢复指定的工作现场，git stash apply stash@{0}

如果想快速同步合并其他分支的某一版本更新内容（特定的提交），可使用git cherry-pick 4c805e2 红色是其他分支修复提交过的ID号，执行完，当前分支也就同样commit了一样的内容修复。

git branch -D feature-vulcan

强制删除没被合并的分支，参数是大写D

git remote

查看远程库信息

git remote –v

查看远程库详细信息

git push origin master

把本地master分支推送到远程仓库

git checkout -b 本地分支 origin/远程分支

创建一个本地分支并且与远程对应的分支同步

git branch --set-upstream-to=origin/dev dev

把本地分支dev与远程库分支dev链接起来，后面才可以git pull

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

同上，跟踪远程分支操作，跟踪了才可以随心所欲git pull拉取分支代码。

git rebase

rebase操作的特点：把分叉的提交历史“整理”成一条直线，看上去更直观。缺点是本地的分叉提交已经被修改过了。

打标签