安装Git

sudo apt-get install git

or

官网下载源码 解压 ./config make sudo make install

创建版本库

mkdir learngit//创建空目录

cd learngit

pwd//显示当前目录

git init//建好仓库 并返回提示信息这是空仓库：Initialized empty Git repository in XXXX目录

ls -ah//显示当前目录下内容（.开头的隐藏文件也会列出）

所有版本控制系统只能跟踪文本文件的改动，如TXT，网页，代码。图片，视频等二进制文件可管理但无法跟踪改动（知道改动，但是不知道改了啥）

将文件加入Git仓库（必须在Git仓库目录内执行）：

git add readme.txt//最后一项为文件名 添加到暂存区

git commit -m "wrote a readme file"//-m后面输入的是本次提交的说明，任意内容

//commit可以一次提交很多文件，即可以git add若干次后使用一次commit。

时光机穿梭

git status//仓库当前状态 可以告诉你有文件被修改过

git diff readme.txt//查看具体修改了什么内容

版本回退

git log//查看提交历史记录

//git log --pretty=oneline可以只显示版本号和提交说明

git reset --hard HEAD^//HEAD表示当前版本 HEAD^是上一个版本 HEAD^^上上 往上100个版本为HEAD~100

//此时再用git log发现最新一次提交记录消失

git reset --hard 1094ad//版本号 可以去到任一版本 版本号可以不打全，Git会自动去找

git reflog//显示每次命令 从而可以在关机后找到版本号

工作区和暂存区

git add实际是将文件修改添加到暂存区 git commit实际就是把暂存区的所有内容提交到当前分支

管理修改

修改后如果直接commit 修改是不会提交的

git diff HEAD -- readme.txt//查看工作区和版本库里面最新版本的区别

撤销修改

git checkout -- readme.txt//把readme.txt文件在工作区的修改全部撤销

回到最近一次的git commit/git add的状态

git reset HEAD readme.txt//撤销暂存区的修改，重新放回工作区

删除文件

git rm test.txt//将工作区文件删除

git status操作后会返回提示

git rm test.txt

git commit -m "remove test.txt"//在版本库中删除该文件

如果发生误删

git checkout -- test.txt//用版本库里的版本替换工作区的版本 即最新一次git add/git commit的版本

远程仓库

git remote add origin git@github.com:cccya//关联远程库

git push -u origin master//第一次推送master分支的所有内容

git push origin master//推送最新修改

git remote -v//查看远程库信息

git remote rm origin//origin为远程库名字

分支管理

git checkout -b dev//创建并切换到dev分支 相当于git branch dev git checkout dev

git branch//查看当前分支 如有dev master 当前分支前有\*号

git merge dev//将dev分支合并到master上

git branch -d dev//删除dev分支

git switch -c dev//创建并切换到新的dev分支

git switch master//切换到已有的master分支

解决冲突

手动解决冲突 再提交

git log --graph命令可以看到分支合并图

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

分支管理策略

禁用Falst forward模式，Git会在merge 时 生成一个新的commit

git merge --no--ff -m "merge with no-ff" dev//-m参数后加上描述

实际开发中，master分支应该仅用来发布新版本，平时不能在上面干活。干活都在dev分支上。到某个时候，把dev分支合并到master上，在master分支发布1.0版本

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合并看不出来曾经做过合并

Bug分支

当前正在dev上进行工作，突然收到一个Bug，于是新建issue-101来修复。但是dev还未完成无法提交，使用stash功能，将当前工作现场“储存”起来

先在dev分支上使用命令git stash

然后回到master分支新建一临时分支issue-101进行Bug修复

完成后回到dev分支

git stash list//查看

git stash apply//恢复原工作，但是stash内容不删除

git stash pop//恢复同时把stash内容也删了

可以多次stash，即先用git stash list 查看到有工作stash@{0} 然后git stash apply stash@{0}

git cherry-pick 4c805e2//复制一个特定的提交到当前分支 4c805e2为commit后反馈的提交信息

Feature分支

对于还未merge的分支 如果要进行合并

git branch -D feature-vulcan//feature-vulcan为分支名 -D表示强制删除

多人协作

git remote//查看远程库信息 加上参数-v显示更详细的信息

一般来说，master是主分支，要时刻与远程保持同步。

dev是开发分支，团队所有成员都需要在上面工作，也需要远程同步。

bug分支只用于本地修复bug，一般情况下无需push

feature分支是否推送到远程，取决于你是否和你的小伙伴在上面开发

抓取分支：

git clone git@github.com:michaelliao/learngit.git//前为远程库的username 后为本地库名字

这样clone下来只能看到本地的master分支，若要在dev上进行开发，就必须创建origin的dev分支到本地

git checkout -b dev origin/dev

由于多人同时开发，push时可能发生冲突。解决办法为先用git pull把最新提交的origin/dev抓取下来，然后在本地合并，解决冲突，再推送

git pull

多人协作工作模式：

1.试图用git push origin <branch-name>推送自己的修改

2.如果推送失败，则因为远程分支比你的本地更新，需要先用git pull试图进行合并

3.如果合并有冲突，则解决冲突，并在本地提交

4.没有冲突或者解决冲突后，用git push origin <branch-name>就能推送成功

Rebase

git rebase//把本地未push的分支提交历史整理成直线

rebase的目的是使我们在查看历史提交的变化时更容易，因为分叉的提交需要三方对比

pull如果有冲突，不要直接手动解决+rebase，否则冲突会恢复 可以将冲突文件add commit之后 rebase一下 再解决+rebase

创建标签

先切换到需要打标签的分支上

git tag v1.0//v1.0为标签名 默认标签打在最新提交的commit上

git tag//查看所有标签 标签按字典序排列

git tag v0.9 f52c623//如果要给过去某次commit打标签，用git log查看历史提交的commit id

git show v0.9//查看标签信息

git tag -a v0.1 -m "verssion 0.1 released" 1094adb//-a指定标签名 -m指定说明文字

操作标签

git tag -d v0.1//删除标签

创建的标签都只储存在本地，不会自动推送到远程。如果要推送

git push origin v1.0//推送某一标签

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

若要删除一个推送到远程的标签 先从本地删除，再从远程删除

git tag -d v0.9

git push origin :refs/tags/v0.9

使用Github

参与一个开源项目：

访问项目主页，点“Fork”就在自己的账号下克隆了与一个仓库，然后在自己的账号下clone

git clone git@github.com:michaelliao/bootstrap.git

可以推送一个pull request给官方仓库来贡献代码

使用Gitee

git remote rm origin//删除远程库

忽略特殊文件

在Git工作区的根目录下创建一个特殊的.gitignore文件，把要忽略的文件名填进去

忽略文件的原则：

1.忽略操作系统自动生成的文件，比如略缩图等；

2.忽略编译生成的中间文件、可执行文件等，也就是如果一个文件是通过另一个文件自动生成的，那么自动生成的文件就没必要放进版本库，比如Java编译产生的.classs文件；

3.忽略你自己的带有敏感信息的配置文件，比如存放口令的配置文件。

若想添加一个文件到Git，但是失败，可能是因为这个文件被忽略了。如果确实想添加该文件，可以用-f强制添加到Git。

git add -f App.class

或者想找出到底是.gitignore哪里写的有问题

git check-ignore -v App.class

在.gitignore中：

#排除所有.开头的隐藏文件

.\*

#排除所有.class的文件

\*.class

#不排除.gitignore和App.class

!.gitignore

!App.class

配置别名

git config --global alias.unstage 'reset HEAD'//--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

//配置后 git unstage test.py=git reset HEAD test.py

配置时加上--global是针对当前用户起作用的，如果不加，那只针对当前仓库起作用

每个仓库的配置文件都放在.git/config文件中

当前用户的Git配置文件放在用户主目录下的一个隐藏文件.gitconfig中