**Git分为4部分：**

工作区（Workspace）：本地文件

缓存区（Index / Stage）：也叫暂存区，一般放在.git文件的index文件下

本地版本库（Repository）：放在本地.git文件夹内的版本库

远程版本库（Remote）：放在远程的版本库

origin ：默认的远程仓库名称，可修改

head：当前活跃分支的游标，可以用 checkout 命令改变 HEAD 指向的位置。HEAD 就总是指向当前所在分支。不可修改

Master: 首次创建仓库时默认分支的名字，在大多数情况下，master是指主干分支，可修改成其他分支，但不可删除





git config --global user.name "yerundong"

git config --global user.email [414868970@qq.com](mailto:414868970@qq.com)

git全局初始化的用户名和邮箱，保存在你的git安装目录下，这两条配置很重要，每次 Git 提交时都会引用这两条信息。--global代表全局的，你所有的项目都会默认使用这里配置的用户信息。去掉 --global，新的设定保存在当前项目的 .git/config 文件里，这是局部的。

$ git remote -v

查看当前远程地址，可能是https或ssh的

https储存密码账号方式：

长期储存：

$ git config --global credential.helper store

设置记住密码（默认15分钟）：

$ git config –global credential.helper cache

如果想自己设置缓存时间，可以这样做（下面为1小时）：

git config credential.helper ‘cache –timeout=3600’

清除缓存

git config --system --unset credential.helper

$ git remote add [option] name(origin) url

添加远程库地址

$ git remote set-url origin [url]

切换远程库地址

$ git init

初始化git仓库，在根目录生成.git文件

$ reset

清屏

$ git clone [url]

把远程仓库仓库克隆到本地当前目录

url可为https或ssh。

以Github为例：

https:

https://github.com/yerundong/myEnd

ssh:

[git@github.com:yerundong/myEnd.git](mailto:git@github.com:yerundong/myEnd.git)

$ git pull

从远程库拉取代码到本地

$ git fetch --all

$ git reset --hard origin/master

$ git pull

git放弃修改，强制覆盖本地代码

$ git add [fileName1 fileName2 …]

把工作区的文件添加到缓存区

$ git add .

添加所有修改的文件

$ git status

查看缓存区可提交的文件

$ git log

查看commit日志

$ git reflog

git reflog 可以查看所有分支的所有操作记录（包括commit和reset的操作）

$ get reset--hard [版本号]

恢复到特定版本

$ git push -f origin <branchName>

强制推送的远程分支

$ vi [fileName]

查看文件

$ git diff

查看工作区与缓存区的文件差异，若git add，则无差异

$ git commit –m“message”

提交缓存区的文件到本地版本库，message是提交信息

$ git commit –am“message”

将所有修改的文件都提交到本地版本库，省略了git add .这个操作

$ git push

将版本仓库推送（同步）到远程仓库

$ git branch

查看所有本地分支

$ git branch -r

查看所有远程分支

$ git branch -a

查看所有本地和远程分支

$ git branch [branchName]

舍弃本地代码,远端版本覆盖本地版本(慎重)

$:git fetch --all

$:git reset --hard origin/master

$:git fetch

创建新的本地分支

$ git checkout [branchName]

切换本地分支

$ git checkout -b [branchName]

新建一个本地分支，并切换到该分支

$ git checkout -b [本地分支] origin/ [远程分支]

在本地创建分支dev并切换到该分支 这个将会自动创建一个新的本地分支，并与指定的远程分支关联起来。

$ git checkout -

切换到上一个分支

$ git checkout -- [fileName1 fileName2 ...]

获取本地仓库代码里的特定文件覆盖工作区，fileName1是本地绝对地址

$ git checkout .

获取本地仓库代码里的所有被修改过的文件覆盖工作区

$ git branch –d [branchName]

删除本地分支

$ git push origin --delete [branchName]

$ git push origin :[branchName]

删除远程分支

$ git push origin [branchName1]:[branchName2]

把本地分支branchName1推送到远程仓库，名为branchName2

如本地远程同名，就这样写：

$ git push origin [branchName]

$ git branch --set-upstream-to=origin/远程分支的名字 本地分支的名字

本地分支和远程分支建立建立追踪关系

$ git branch –vv

查看是否建立联系

$ git merge [branchName]

合并指定分支到当前分支