**###ssh key及 配置信息#############**

设置Git的user name和email：

$ git config --global user.name "memory.qiu"

$ git config --global user.email "qiuhanjiang1986@163.com"

生成SSH密钥过程：

1.查看是否已经有了ssh密钥：cd ~/.ssh

如果没有密钥则不会有此文件夹，有则备份删除

2.生存密钥：$ ssh-keygen -t rsa -C “haiyan.xu.vip@gmail.com”

按3个回车，密码为空

.在github上添加ssh密钥，这要添加的是“id\_rsa.pub”里面的公钥

###本地初始化一个**目录**为代码仓库##################

git init 初始化仓库

1、创建一个新目录作为代码仓库

2、执行 git init；

执行之后会发现多了一个.git目录

这就说明仓库已经初始完成

3、在github上创建一个新的代码仓库（创建对应的代码仓库）

4、git remote add origin git@github.com:memoryqiu/memory2017.git

5、正常提交代码即可。

###克隆一个项目#######

https 速度慢、需要输入账号和密码/ ssh 会被公司限制 两种协议

要克隆一个仓库，首先必须知道仓库的地址，后使用git clone命令克隆然。

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

Git支持多种协议，包括https，但通过ssh支持的原生git协议速度最快。

git clone https://github.com/memoryqiu/memory2015.git

or

git clone git@github.com:memoryqiu/memory2015.git

通常推荐大家直接git clone 一个项目的方式来创建本地的代码仓库！

演示一下在github上如何创建一个代码仓库！！！

###创建分支################

Git鼓励大量使用分支：

创建分支：git branch <name> （默认当前所在分支为基础分支，可以指定其它分支）

切换分支：git checkout <name>

创建+切换分支：git checkout -b <name> <基础分支名称>（origin/master）

提交刚创建的分支：git push origin branch-name

更新分支：Git pull origin branchname or Git fetch origin branchname git merge --no-ff 目标分支

合并某分支到当前分支：git merge --no-ff <branch1分支名称>（把分支branch1合并到当前分支上）

git merege (fast forward and --no-ff )

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

请注意--no-ff参数，表示禁用Fast forward请注意--no-ff参数，表示禁用Fast forward

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

因为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

查看本地分支：git branch or git branch -l

查看远程分支：git branch -r

查看所有分支（本地+远程）：git branch -a

带\*的分支为当前工作的分支

删除本地分支：

git branch -d branchnamne

or

git branch -D branchname 强制删除

删除远程分支：

Git push origin --delete branchname (删除远端分支branchname)

Git push origin :branchname (直接删除远端分支branchname)

获取远程仓库的分支（本地没有的分支）

git pull origin $branch:$branch

实例：git pull origin branchname:branchname

###git commit####################

git commit -m'' 只会提交已经git add过的文件（包括新增、修改、删除等所有修改操作）

git commit -a -m'' 会提交除新增未add的文件之外的所有修改操作

###git diff ####################

Git diff filename 显示本地文件修改的内容同版本库的差别

Git diff branchone branchtwo 对比两个branch不同之处

Git diff tagone tagtwo 对比两个tag 不同之处

###删除文件###########

在本地直接删除文件

1、rm file （工作区中文件删除）

2、git add file （文件删除的状态添加到暂存区中）

3、git commit （提交文件删除的状态到本地的版本库中）

以上三步可以缩减为两步：

1、git rm file （直接把删除的文件添加到暂存区中）

2、git commit （提交文件删除的状态到本地的版本库中）

###删除文件###############

1、git rm filename 然后 git commit 即可

2、rm filename / git add / git commit

git rm filename 就是 rm filename 的简化版

###查看当前工作区修改状态#######

git status;

要随时掌握工作区的状态，

使用git status命令

如果git status告诉你有文件被修改过，

用git diff可以查看修改内容。

git diff 默认查看所有被修改过的文件（不包括新增未add的文件）

git diff filename (查看具体某个文件的修改内容 )

###查看log记录#####

git log 默认列出之前提交过记录（根据提交时间由近到远排列）

git log -n 列出指定提交次数的记录

git log --pretty=oneline 列出简单记录信息（commitid 和 提交注释）

git reflog -n 查看指定次数的提交操作的记录信息（commitid）

log and reflog

两者都是查看所有的commit操作的记录，log查看不到reset回退时跳过的commit操作的commitid

（可以夸版本回退一次，进行实际演示）

git log

git reflog

git reset --hard commitid

git log

git reflog

git log --graph 可以查看分支合并图

###撤回和回滚#######

1、在工作区内修改了文件但没有提交到暂存区撤销所有修改：

git checkout -- file.name 可以丢弃工作区的修改 ，只针对已跟踪的文件

git checkout -- file 命令中的--很重要，没有--，就变成了“切换到另一个分支”的命令

2、文件已经添加到暂存区撤回：

git reset HEAD file

or

git rm --cached filename

3、文件已经提交到了本地的版本库且未推向远端的版本库中：

git reset --hard HEAD~n

or

git reset --hard commitid(推荐)

HEAD 表示提交的最新版本。HEAD^表示上一个版本，HEAD^^表示上上个版本。HEAD~100表示往上100个版本。

(上两个版本 HEAD~2)

4、文件已经从本地仓库推倒远程仓库的回退

git log or git reflog 找到目标版本

git reset --hard commitid

git push -f origin master (-f 参数是强制覆盖的意思，没有该参数会推送失败，因为远程仓库的版本高于当前版本)

git revert commitid

git revert 是生成一个新的提交来撤销某次提交，此次提交之前的commit都会被保留

git reset 是回到某次提交，提交及之前的commit都会被保留，但是此次之后的修改都会被退回到暂存区

团队开发时，版本回退一定要通知其他人员！如果其他同事在每次提交前没有先更细后提交的意识或习惯的话，回退是白费功夫。

http://yijiebuyi.com/blog/8f985d539566d0bf3b804df6be4e0c90.html

###状态冻结####################

git stash 冻结当前修改状态

git stash list 列出总结列表

git stash pop 可以按版本恢复某个冻结状态并删除

git stash apply 只是按某个版本恢复某个冻结状态不删除

git stash drop 删除指定某个冻结状态

###tag管理#####################

创建tag:

git tag -a tagname -m “添加tag的描述信息” (tag要有个规则标识例如：feature\_abc\_v1.1 ，给当前分支打一个tag名为tagname)

命令git tag <name>用于新建一个标签，默认为HEAD，也可以指定一个commit id；

提交创建的tag:

git push origin tagname (同提交分支操作一样)

命令git tag可以查看所有标签。

查看tag：

git tag

or

git tag -l

查看标签的具体信息：

git show tag-name

删除标签：

Git tag -d tagname (本地删除)

Git push orign (空格):refs/tags/tagname (把删除的tag提交到远端)

删除标签实例：

git tag -d v0.9

git push origin :refs/tags/v0.9

获取远程仓库的tag信息（本地没有）

git fetch origin tag $tag

###同远程仓库关联#########

查看当前代码关联的远程仓库信息

git remote 直接回车，列出所有关联的远程仓库

git remote -v 回车 列出每个远程仓库名及对应的仓库地址url

git remote add [shortname] [url]: 添加同一个远程仓库的关联关系

git remote remove shortname : 删除同一个远程仓库的关联关系

要关联一个远程库，使用命令git remote add origin git@server-name:path/repo-name.git；

关联后，使用命令git push -u origin master第一次推送master分支的所有内容；

此后，每次本地提交后，只要有必要，就可以使用命令git push origin master推送最新修改；

分布式版本系统的最大好处之一是在本地工作完全不需要考虑远程库的存在，也就是有没有联网都可以正常工作，而SVN在没有联网的时候是拒绝干活的！

当有网络的时候，再把本地提交推送一下就完成了同步，真是太方便了！

　.gitignore规则不生效的解决办法

把某些目录或文件加入忽略规则，按照上述方法定义后发现并未生效，原因是.gitignore只能忽略那些原来没有被追踪的文件，如果某些文件已经被纳入了版本管理中，则修改.gitignore是无效的。那么解决方法就是先把本地缓存删除（改变成未被追踪状态），然后再提交：

git rm -r --cached .

git add .

git commit -m 'update .gitignore'

####分支覆盖 强推#############

[git将一个分支的内容替换为另一分支内容](https://www.cnblogs.com/goldlone/p/8420636.html)

假设我想将我的linux分支内容替换master分支的内容。

# 切换到master分支

git checkout master

# 再将本地的master分支重置成linux

git reset --hard linux

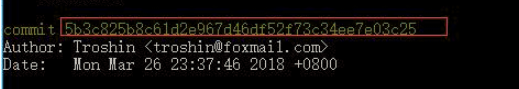
# 最后推送到远程仓库master分支

git push origin master --force

####本地分支回滚#########

git本地版本回退

Git reset --hard commit\_id(可用 git log 查看)



####拉取远程分支并创建到一个本地分支

git branch -r

git checkout -b fenzhi001 origin/fenzhi001

**git checkout -b 本地分支名 origin/远程分支名**

### 合并远程分支到本地 #######

Git pull origin 远程分支