配置Git:

git config --global user.name "your name"

git config --global user.email "Gitemail@xxx.com"

通过git config --list 查看配置信息

1 通过命令 git init 把这个目录变成git可以管理的仓库

2 使用命令 git add 文件名 添加到暂存区里面去

3. 用命令 git commit -m "提交注释" 告诉Git，把文件提交到仓库

4 用命令git status 查看文件状态

5 用命令git diff 文件名 查看修改

6 git log命令显示从最近到最远的显示日志

7 git reset --hard HEAD^  把当前的版本回退到上一个版本（^决定倒退版本数）

8 git reflog 获取到版本号

9 git reset --hard 版本号 恢复到指定版本

10 git checkout --文件名 把该文件在工作区做的修改全部撤销

1.文件自动修改后，还没有放到暂存区，使用 撤销修改就回到和版本库一模一样的状态

2.另外一种是文件已经放入暂存区，接着又作了修改，撤销修改就回到添加暂存区后的状态。

git checkout -- 文件名 撤销命令让其回到暂存区后

删除文件：

1 rm 文件名后 执行commit提交命令彻底删除

2 rm 文件名后 未执行commit提交命令 可通过 git checkout -- 文件名 恢复文件

远程仓库：

1.创建SSH Key

ssh-keygen -t rsa –C “youremail@example.com”

2.登录github 黏贴id\_rsa.pub

把本地仓库的内容推送到GitHub仓库

1.git remote add origin <https://github.com........>

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。即：

2.git push -u origin master

之后提交修改只需要通过

3.git push origin master

创建与合并分支：（dev为创建的一个分支）

1.创建分支： git branch dev

2.切换分支： git checkout dev

创建切换分支： git checkout –b name

3.查看分支： git branch

4.合并分支： git merge dev（在master下执行则把dev 分支合并到master分支上）

5.删除分支： git branch -d dev

多人协作：

1.要查看远程库的信息 git remote

2.查看远程库的详细信息 git remote –v

现在我们的小伙伴要在dev分支上做开发，就必须把远程的origin的dev分支到本地来，于是可以使用命令创建本地dev分支：git checkout –b dev origin/dev

git pull把最新的提交从origin/dev抓下来，然后在本地合并，解决冲突，再推送。

git pull也失败了，原因是没有指定本地dev分支与远程origin/dev分支的链接，根据提示，设置dev和origin/dev的链接：如下：

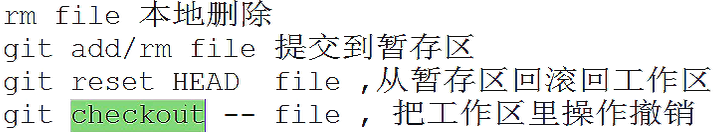
git branch --set-upstream dev origin/dev

这回git pull成功，但是合并有冲突，需要手动解决，解决的方法和分支管理中的 解决冲突完全一样。解决后，提交，再push

cat 文件名 并解决冲突

最后git push origin dev

1.关于checkout(回滚)



nginx基本配置：

1.通过 service --status-all查看是否装好nginx

2.配置文件目录/usr/local/nginx/conf/nginx.conf,配置文件的修改需要通过vi进行修改

修改default配置文件

3.sudo cat default

**3.罗列出与nginx相关的软件**

dpkg --get-selections|grep nginx

**5.查看nginx正在运行的进程，如果有就kill掉**

ps -ef |grep nginx

4.利用rz把本地资源上传到虚拟机