## Git常见命令和操作

# git 是一个文件管理工具，主要用来管理工作文件。

git init：初始化一个仓库~也就是创建了一个仓库

git status：查看工作状态

git add 文件名：将文件的改动添加到暂存区

git commit –m 描述：将提交到暂存区的文件改动更新到本地仓库，注意描述中不能有空格

=》用户的创建

全局用户创建：

git config --global user.email 邮箱地址：配置全局用户的邮箱

git config --global user.name 用户名：配置全局用户的用户名

局部用户创建：

git config user.email 邮箱地址：配置局部用户的邮箱

git config user.name 用户名：配置局部用户的用户名

# 一般情况下我们都会使用全局用户，让所有的本地文件同属于一个用户

# 全局用户可以在C:\Users\Administrator/.gitconfig中查看

# 局部用户可以在所属.git文件中config查看

git log：查看工作日志，只会显示当前版本之前版本的提交的操作

git reflog：查看全部工作信息，包含了版本前后的所有提交，版本更替、跳转等操作

git checkout -- 文件名：会先从暂存区获取版本，如果上一个版本已经提交本地仓库，会从本地仓库中获取版本信息，然后将工作区的信息撤回到上一次提交的版本。注意：如果要进行撤回操作的话，一定不要把当前要撤回的操作增加到暂存区，因为系统会有限从暂存区获取版本，修改会不成功。

git reset HEAD 文件名：将增加到暂存区的版本从暂存区移除，如果工作区又对其作出了新的修改，以工作区为准。

git reset --hard 版本号：回到指定的版本~工作区

git reset --hard HEAD^：回到当前版本的上一个版本，一个^表示上跳一个版本，~n回到上n个版本。

git diff：比较当前工作区域与本地仓库的内容的差别，系统还是会优先查找暂存区，如果暂存区有相同的文件版本的话会比较当前工作区域与暂存区相同文件的版本差别。如果有暂存区不存在的文件系统会到本地仓库查找，给出最终对比结果。

git branch：罗列当前区域的所有分支

git branch 分支名：创建一个分支

git checkout 分支名：切换到指定的分支

git checkout –b 分支名：创建并切换到新分支

git merge 分支名：合并分支到当前分支

git branch –d 分支名：删除分支

# 分支创建后会拷贝当前所在分支的所有文件，系统默认有一个分支master，我们一般情况下会将其当做主分支，但是分支是不分主次的，任何一个分支都可以当做主分支。只是实际开发中规范管理的要求。

# 新创建的分支内的各种操作不会影响其他分支，但是如果对主分支已有的文件进行操作的话，可能会造成内容冲突。因为我们分支最后都是要合并到主分支当中的。在合并的时候，对于相同的文件，系统会按照行数逐行合并。这样，如果两个分支的这个文件中的某一行出现了差别，在合并的时候会出现错误。包括空行不一致也会出现错误。

# 我们实际操作时如果碰到这种问题的话，我们需要与修改者进行沟通，商议修改方法，在主分支内修改后提交。

# 如果我们在一个分支的工作区域做出了修改，没有提交，哪怕是存在暂存区，就切换到其他分支，工作区和暂存区的修改内容就会影响到其他分支，如果在其他分支中进行删除的话，修改的分支也会删除操作。所以我们在进行分支跳转的时候一定要保证当前分支工作信息处于clean状态。

==》远程仓库操作

git clone 仓库地址：克隆一个仓库

git push [origin 分支名称]：提交本地文件到远程仓库，默认是master分支。如果有需要的话可以通过origin 分支名称来选择提交的分支。

# 实际项目中一般不会直接提交到主分支，所以我们可以使用git push origin dev:dev 命令创建dev分支，提交当前dev分支

git pull：拉取远程仓库更新文件。

# pull一般只做远程文件更新时候的拉取新信息的操作，不会做本地文件的恢复操作，本地文件如果删除的话，可以在本地文件记录中查找恢复。可以强行理解为pull会查找本地日志版本，如果存在版本与远程仓库版本一致，则不会做任何拉取。

# git 中HEAD关键字的含义可以理解为当前分支的意思。