```
git init #初始化项目
git add 文件名 #添加指定文件
git add . #添加所以文件
git commit -m "提交注释" #添加一个版本
git log #查看历史版本（不包含回退部分的）
git reflog #查看历史版本（详细 ， 包含回退部分的版本）
git reset --hard 提交的版本号 #回退到指定的历史版本
git check out #

 git reset --soft
 git reset head
 git reset --mix

 gitlab 自己创建代码托管（基于linux）
```

分支操作

```
# 删除远程分支
git branch -a #查看已有的本地及远程分支
git push origin --delete dev #删除远程分支
git branch -d dev #删除本地分支

git remote add origin ..... # 添加git地址
git push origin dev 推送到线上dev分支
git pull origin dev 获取线上的dev分支

 提交的代码晚于github行的代码，会有错误提示，叫你重新拉取代码
 拉取代码会提示你合并代码

 git pull origin dev = git fetch origin dev
  + git merge origin/dev


```

```
git pull 的代码会有分叉（有忘记提交的代码的情况：比如在公司开的代码已经commit 但是忘提交，回家继续开发提交，第二天到都公司pull代码就会出现分叉）

下面的方法不会有分叉
git fetch origin dev
git rebase origin/dev #rebase 保持提交记录的整洁
```

开发过程中临时需要修复bug 或者 临时新需求

```
方式一：
# git stath作用，帮助我们暂时存储已经开发的一些功能代码，继续做其他事情，做完之后，再回来继续开发
git stash #找到当前所有的红色文件，临时删除并存储到“某个地方”（用于功能开发到一半时，要解决之前的问题 ）
git stash list # 查看“某个地方”存储的所有记录
git stash clear #清空"某个地方"
git stash pop #将第一条记录从"某个地方"重新拿回工作区（可能会有冲突）
git stash apply 编号 #将指定编号从"某个地方"重新拿到给工作区（可能会有冲突）
git stash drop 编号 #删除制定的编号记录

方式二：
git branch dev 拷贝当前内容新建一个dev的分支
git branch 显示当分支列表
git checkout dev 切换到dev分支
开发...
git merge dev # 在master 分支下执行改代码，将dev 合并到master下
git branch -d dev #删除 dev 分支

注意：在公司不要用master分支下修改代码，要创建新的分支进行修改，修改完成后再合并

```

git的命令行输出正确地显示中文文件名

```
git config --global core.quotepath off
```

不用重复输入用户名和密码的方法

https：

```
git remote add origin https://用户名：密码@ git地址
```

ssh

.gitignore