git remote -v 查看远程链接库

ssh -T 查看SSH链接状态

git rm -rf --cached 文件名 删除github上的文件夹但不删除本地

git commit

git push

git config --list查看config配置

git remote rm origin删除远程库

git checkout -b <branchname> 创建一个本地分支

git checkout master 切换回主分支

git branch -d <branchname> 删除一个本地分支

git push origin <branchname>

git diff <source\_branch> <target\_branch>

git push origin <branchname> 推送分支

git remote add origin <server> 添加远程库

git clone <远程地址> clone

一、远程仓库有master和dev分支  
1. 克隆代码

git clone https://github.com/master-dev.git

# 这个git路径是无效的，示例而已

2. 查看所有分支

git branch --all

# 默认有了dev和master分支，所以会看到如下三个分支

# master[本地主分支] origin/master[远程主分支] origin/dev[远程开发分支]

# 新克隆下来的代码默认master和origin/master是关联的，也就是他们的代码保持同步

# 但是origin/dev分支在本地没有任何的关联，所以我们无法在那里开发

3. 创建本地关联origin/dev的分支

git checkout dev origin/dev

# 创建本地分支dev，并且和远程origin/dev分支关联，本地dev分支的初始代码和远程的dev分支代码一样

4. 切换到dev分支进行开发

git checkout dev # 这个是切换到dev分支，然后就是常规的开发

5. 为了更好的理解，最好继续看看下文。  
  
二、假设远程仓库只有mater分支  
1. 克隆代码

git clone https://github.com/master-dev.git

# 这个git路径是无效的，示例而已

2. 查看所有分支

git branch --all

# 默认只有master分支，所以会看到如下两个分支

# master[本地主分支] origin/master[远程主分支]

# 新克隆下来的代码默认master和origin/master是关联的，也就是他们的代码保持同步

3. 创建本地新的dev分支

git branch dev # 创建本地分支

git branch # 查看分支

# 这是会看到master和dev，而且master上会有一个星号

# 这个时候dev是一个本地分支，远程仓库不知道它的存在

# 本地分支可以不同步到远程仓库，我们可以在dev开发，然后merge到master，使用master同步代码，当然也可以同步

4. 发布dev分支  
发布dev分支指的是同步dev分支的代码到远程服务器

git push origin dev:dev # 这样远程仓库也有一个dev分支了

5. 在dev分支开发代码

git checkout dev # 切换到dev分支进行开发

# 开发代码之后，我们有两个选择

# 第一个：如果功能开发完成了，可以合并主分支

git checkout master # 切换到主分支

git merge dev # 把dev分支的更改和master合并

git push # 提交主分支代码远程

git checkout dev # 切换到dev远程分支

git push # 提交dev分支到远程

# 第二个：如果功能没有完成，可以直接推送

git push # 提交到dev远程分支

# 注意：在分支切换之前最好先commit全部的改变，除非你真的知道自己在做什么

6. 删除分支

git push origin :dev # 删除远程dev分支，危险命令哦

# 下面两条是删除本地分支

git checkout master # 切换到master分支

git branch -d dev # 删除本地dev分支

撤销commit:

先查看commit 日志

git log

git revert <hash-id>

撤销add:

git reset <filename>

如:

Git reset . //撤销当前目录下所有add文件