#### Git 常用指令

1. 生成ssh
2. 拉去代码：

Git clone url

Gir clone url -b 分支

1. 切换到远程分支
   1. git branch -a 查看所有分支
   2. git checkout -b 分支名 origin/分支名
2. 远程地址设置相关

git remote -v 查看远程地址

git remote origin 远程地址 设置远程地址

git push origin 分支名:推送到远程分支 推送本地分支到远程  
git push origin :推送到远程分支 删除远程分支

git remote set-url origin 新地址 修改远程地址

1. stash
   1. git stash 暂存当前工作空间未提交的操作
   2. git stash list 查看stash列表
   3. git stash pop 恢复暂存内容到当前工作空间
2. 撤销上次commit
   1. git log 查看提交日志
   2. git reset --hard commitNo 重置到指定提交id
3. 修改本地分支对应的远程分支
   1. git branch --set-upstream-to=origin/远程分支名

# 当前在 dev\_duanwu 分支，使用 git pull 拉取远程时，一大堆冲突

git pull origin dev\_duanwu

# 解决方法

# 1. 取消当前冲突比较多的合并

git merge --abort

# 2. 使用 fetch 更新本地的  origin/dev\_duanwu 分支

git fetch origin dev\_duanwu

# 3. 合并时带上参数，忽略只有空格（换行符）的改动

git merge -Xignore-all-space origin/dev\_duanwu

# 4. 此时，冲突一般很少，或者没有；如果有的话，说明不是空格引起的冲突，只能手动解决冲突

# 5. 解决完冲突

git add .

git commit -m "merge origin/dev\_duanwu"

git push origin dev\_duanwu

本地比较两个分支

|  |
| --- |
| # -w 忽略空格导致的差异  git diff prod..dev\_xxx\_20190523 -w  # 比较两个文件  git diff prod..dev\_xxx\_20190523 -w 工程目录/src/main/java/com/szy/skill/thread/TestGit.java    # 如果本地要merge两个分支，最好也带上忽略空格  git checkout dev\_xxx\_20190523  git fetch origin prod  git merge -Xignore-all-space origin/prod |

## 将单个文件回退到指定版本

# 1. 进入到文件所在文件目录，或者能找到文件的路径查看文件的修改记录

git log TestGit.java

# 修改详情

git log -p TestGit.java

# 2. 回退到指定的版本

git checkout a4e215234aa4927c85693dca7b68e9976948a35e TestGit.java

# 3. commit checkout下来的版本

git commit -m "revert to previous version"

# 4. 推送到远程仓库

git push origin hotfix\_null

## 把其他分支的commit，加到当前分支

# 使用场景

# 1，提交错分支

# 2，不小心把别人的代码合到了自己的开发分支，自己的开发分支需要重新创建，然后把需要的commit加过来

# 步骤

# 1，找到commit id，命令行不好看的可以用可视化工具SourceTree

git checkout commit所在分支

git log

# 2，切换到需要增加这些commit的分支

git checkout target\_branch

# 3，cherry pick

git cherry-pick commit\_id1 commit\_id2 commit\_id3

# 如果commit比较多，且是连续的

# 不包含commit\_id1

git cherry-pick commit\_id1..commit\_id9

# 包含commit\_id1

git cherry-pick commit\_id1^..commit\_id9

#### Git提交注释规范

**提交格式**

<type>(<scope>): <subject>

// 空一行

<body>

**说明**

type（必需）、scope（可选）和subject（必需）。

<body>(可选)

type用于说明 commit 的类别，只允许使用下面8个标识。

· br: 此项特别针对bug号，用于向测试反馈bug列表的bug修改情况

· feat：新功能（feature）

· fix：修补bug

· docs：文档（documentation）

· style： 格式（不影响代码运行的变动）

· refactor：重构（即不是新增功能，也不是修改bug的代码变动）

· test：增加测试

· chore：构建过程或辅助工具的变动

· revert: feat(pencil): add 'graphiteWidth' option (撤销之前的commit)

scope  
scope用于说明 commit 影响的范围，比如数据层、控制层、视图层等等，视项目不同而不同。

subject  
subject是 commit 目的的简短描述，不超过50个字符。  
以动词开头，使用第一人称现在时，比如change，而不是changed或changes  
第一个字母小写  
结尾不加句号（.）

Body 部分是对本次 commit 的详细描述，可以分成多行。