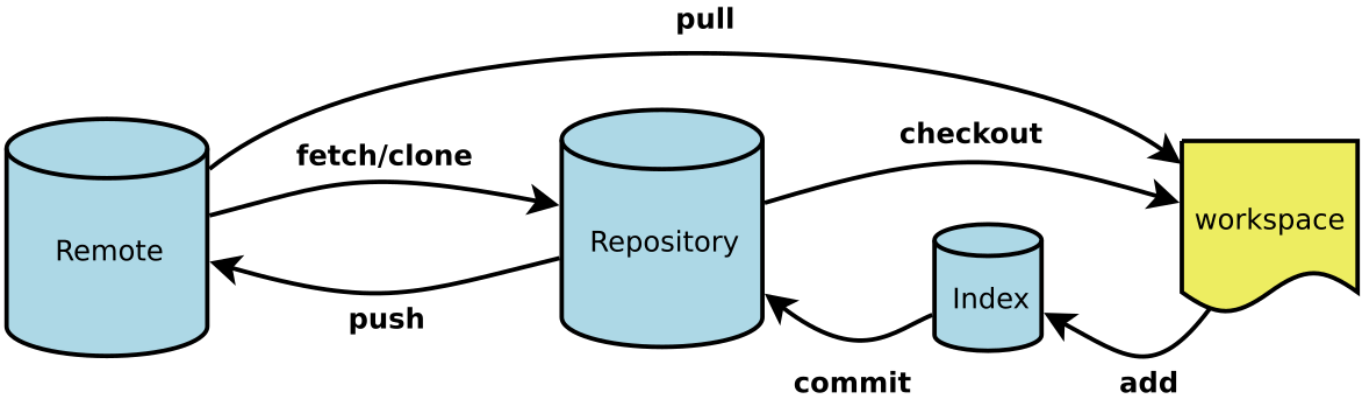
# Git

## Git介绍

Git分为四个部分：

- Workspace：工作区
- Index / Stage：暂存区
- Repository：仓库区（或本地仓库）
- Remote：远程仓库




## Git相关命令

### 初始化配置

```
git init 初始化
git config
    设置系统用户级别签名
        git config --global user.name "jefferson"
        git config -- global user.email "jefferson@163.com"
        配置文件：项目目录下的./.git/config文件
    设置项目级别签名(项目级别高于系统用户级别)：
        git config user.name "jeff"
        git config user.email "jeff@163.com"
        配置文件：家目录下的gitconfig文件
    git config --list 查看所有信息
    git config --global core.quotepath false   应对中文无法显示
```

### 工作区、暂存区、本地仓库的操作

```
git status  查看工作去和暂存区的状态
git add     将工作区的新建和修改提交到暂存区
git commit
    git commit -m "commit message" 文件名 将暂存区的文件提交到本地仓库(-m 可以避免
进入vim，直接添加备注信息)
    git commit -am "commit message" 文件名  将工作区的文件直接提交到本地仓库
```

```
git rm --cache 文件名    从缓存中删除
git checkout 文件名 从缓存中删除就文件的更新版本
```

本地仓库的版本信息查询

```
git log
    git log 提交到本地仓库的历史
    git log --pretty=oneline     以优美的形式显示，每条日志信息只显示一条
    git log --oneline    以优美的形式显示，每条日志信息只显示一条
git reflog  显示日志，显示指针回退的步数
```

本地仓库的前进和回退

```
git reset(hard)
    git reset --hard:
        在本地库移动HEAD指针
        重置暂存区
        重置工作区
        如果只是在暂存区和工作区删除，并没有提交到本地库。可以使用git reset --hard
HEAD，回退到赏赐commit的版本

        git reset --hard 局部索引值
        git reset --hard HEAD~n
        git reset --hard HEAD^^^

    git reset --soft
        只是在本地库中移动

    git reset --mixed
        在本地仓库移动HEAD指针
        重置暂存区
git diff
    git diff 文件名 将工作区和暂存区的文件进行对比
    git diff 历史版本局部索引值 文件名   将历史版本和本地仓库进行对比
```

分支

```
git branch
    git branch name 添加branch的名字
    git branch -v   查看本地分支
    git branch -a   查看本地远程的所有分支
    git branch -r   查看远程的所有分支
    git branch -d   删除本地分支
    git branch -d -r    删除远程分支
    git branch -m oldBranch newBranch   重命名本地分支
    注：
```

```
        在版本控制过程中，使用多条线同时推进多个任务。
        同时并行推进多个功能开发，提高开发效率。
        各个分支在开发过程中，如果某一个分支开发失败，不会对其他分支有任何影响。失败 的分支删除重新开始即可。
        并行开发提高效率，分支开发失败对其他没有什么影响
        例如：master、feature_blue(独立开发的分支，成熟以后再合并)
git checkout name   切换到相应的分支
git merge branchname    合并分支(首先要切换到主体分支，然后合并其他的分支， 分支冲突可以手动删除)
git fetch 链接别名 分支名    获取分支的最新更新
    git log -p FETCH_HEAD   查看取回的数据
    git merge FETCH_HEAD    将获取的最新信息合并
```

远程仓库

```
git remote
    git remote add 别名 链接    为链接取别名
    git remote -v 查看本项目设置的所有链接的别名
git push 链接别名 分支名    将本地仓库传输到远程仓库的相应分支
git pull 链接别名 分支名    将远程仓库内的新版本更新到本地仓库
    git pull = git fetch + git merge
    git fetch 链接别名 分支名   //从远程主机的master分支拉取最新内容
    git merge FETCH_HEAD    //将拉取下来的最新内容合并到当前所在的分支中
git clone 链接名    把远程库完整地下载到本地，创建origin远程地址别名，初始化本地库
```

# .gitignore文件

简介：

```
.gitignore主要用于过滤文件，不用push到github的远程库中。
```

文件内容：

```
# Built application files
*.apk
*.aar
*.ap_
*.aab

# Java class files
*.class

# Folder
/bin/

# File
/bin/start.cpp
```

配置语法：

```
以/开头：表示目录
*：通配多个字符
?：通配单个字符
[]：单个字符的匹配列表
!：不忽略某个文件或者目录
注：
    /folder/ 忽略folder文件夹
    *.cpp   忽略cpp结尾的文件
    start.? 忽略start. 文件
    start.[a, b, c] 忽略start.a, start.b, start.c 文件
    !start.c    不忽略start.c文件
```