Repo简介

尚武 2021-08-09 1,622 阅读5分钟

什么是Repo?

Repo是谷歌提供的一套Python脚本，它封装一系列git命令，用来管理多个git库（最典型的的就是Android）作为git仓库大总管，repo并不复杂，本质上还是git那套东西。

下载安装repo

可以借阅google官网：source.android.com/source/down...

如今已经有了中文版本，官网上介绍了一系列的Android协作开发流程，有必要好好看看。按照流程执行下来就可以了。大体上下载Android代码一套流程就是：

> curl repo > repo init > repo sync

相关介绍

- curl storage.googleapis.com/git-repo-do... > ~/bin/repo
  - 这里只是下载了一个repo引导脚本，不是真正的repo。
- repo init -u android.googlesource.com/platform/ma... -b android-4.0.1_r1
  - repo init执行了获取repo库操作，该步下载真正的repo命令要使用的脚本库。
  - 注意：repo会从本级目录和上级目录中查找.repo文件夹，有则执行main.py，没有则创建文件夹进行初始化。所以同级和父级不可以有.repo文件夹，否则repo会失败。

一些选项

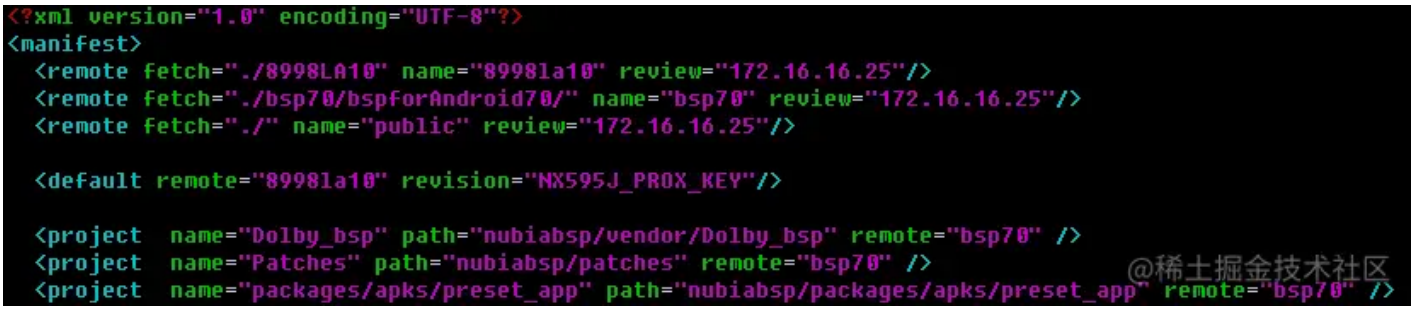
- -u：指定一个URL，其连接到一个maniest仓库，下载AOSP项目必须指定。
- -m：在manifest仓库中选择一个xml文件，可以不写repo init完可以自己再指定。
- -b：选择一个maniest仓库中的一个特殊的分支，注意：分支指的是manifest文件夹那个git库的分支。manifest库一般分支会用来服务于每日构建或者发行版本等为其拉出分支。
- --repo-url：指定repo的地址。

在repo init执行完成之后，当前路径会有一个`.repo`的文件夹（隐藏的）：

目录结构：

- manifests：manifests.git的working dir，分支由-b指定。
- manifests.git：-u 参数指明的清单git库。
- manifest.xml -> xxx.xml：符号链接，指向实际工作的xml，即-m参数的指向。
- project.list：git project清单，每行一个project的所对应的path。
- projects：git库地址，所有projects的git库实际都clone到这里。
- repo：--repo-url指明的repo git库clone到这里。存储所有的repo命令的py脚本。

清单文件介绍



清单文件指示这Android的多个git库的信息清单。

- <remote> 元素： 设置git服务器属性，下面project元素的remote肯定有该元素某条与之对应
  - name：git 服务器的名字
  - alias：git 服务器的别名，一个manifest中name不可以重名，但是alias可以
  - fetch：所有项目的git URL前缀
  - review：指定gerrit服务器
- <default>元素：设定所有的projects的默认属性值，如果project元素没有指定一个属性就用default元素的属性值。相当于默认参数
  - remote：指定上方那些<remote>元素的某一个的name值
  - revision：git的分支
  - sync_j：repo sync 默认并行数
  - sync_c：设置true只同步指定分支
  - sync_s：设为true会同步git的子项目
- <manifest-server>：整个xml中只能存在一个这元素，其url属性用来指定manifest服务的url。不过这个参数不怎么常用。
- <project>：指定一个需要clone的git库

    - name：git库名称
    - patch：同步下来后的本地目录
    - remote：指定之前<remote>元素的某个name
    - revision：指定获取的git 提交点
    - groups：project所属组
- <include>：作用是引入另一个manifest文件。

切换XML

`ln -sf manifests/xxxx.xml manifest.xml` 或者 `repo init -m xxxx.xml`

如果repo init的时候没有指定可以在这里修改xml指向，这里的xml都是项目的一个快照，根据需要将manifest.xml指向你需要的xml即可，xml都在manifests文件夹下。

repo sync -j8

-j8:多线程下载方式，数字根据你处理器核心数。

repo init后的第一次sync，克隆所有的git库到.repo/projects目录下，然后根据xml签出到执行路径：repo sync是将远程的代码先下载到了.repo/projects/**的git库中，然后再一个个checkout到本地上。所以其实一般来说repo sync后会大小比项目实际的要大很多，.repo目录一般会有几十个G，嫌大的话可以根据你实际情况直接删除。

如果不是第一次sync，则repo sync = git fetch，然后根据xml进行checkout（no branch情况下），或者进行rebase操作（存在关联远程分支的本地工作分支）。 如果本地有修改，repo sync可能会失败，远程的修改点和自己本地的修改点冲突（都修改了同一个文件）。就是说repo sync会更新远程最新的代码并rebase到你的代码中。

Repo工作命令

- repo help：什么命令都可以忘，这个不能忘
- repo manifest：生成清单文件即快照。
    - 快照保存你项目当前的一个状态：每个git库的当前跟踪的远程分支，最后提交commit点等等信息。
    - -r xxxx：生成快照 :程序员应该经常使用，就像给git库不断创建分支、打tag一样，每天给工作代码创建一个快照是个好的习惯。

    - 示例：repo manifest –o snapshot.xml -r
- Repo grep：本代码库进行查询，原理和grep一样。
- repo start branch_name --all
    - 指定project (no branch下会建立分支并关联远程分支，但是git status的时候不会显示与远程分支间的差异)
- repo start
    - 这个命令和git checkout –b是有区别的
        - git checkout –b 是在当前所在的分支的基础上创建分支。
        - repo start是在清单文件设定分支的基础上创建特性分支。该分支是属于manifest设定分支的子分支。
        - `repo forall -c 'git checkout -b branch_name $REPO_REMOTE/branch_name'` 也是对全部git库拉出并切换到分支上，加上远程分支还可以关联远程分支，当本地和远程有差异的时候，git status都会提示差异。
- repo checkout branch [project]：切换分支
- repo branches [project]：查看分支
- repo diff [project]：比较差异
- repo stage [project]：git add封装，将文件加入暂存区
    - -i 提供界面供用户选择
- repo prune [project]：删除已经合并的分支
- repo abandon branch [project]：删除指定的分支
- repo status [project]：查看文件状态
- repo push <remotename> [--all |<project>…]：repo会自己查询需要向服务器提交的项目并提示用户。
- repo forall [<project>…] –c <command>：迭代器，可以在所有指定的项目中执行同一个shell指令
- repo forall –c ‘git remote add orgin ssh://justinzhao@127.0.0.1/$REPO_PROJECT.git’：引用环境变量REPO_PROJECT添加远程仓库。
- repo grep：查询工具
- repo version：repo的版本
- repo selfupdate：用于repo自身的更新
- repo upload：推送到gerrit服务器上

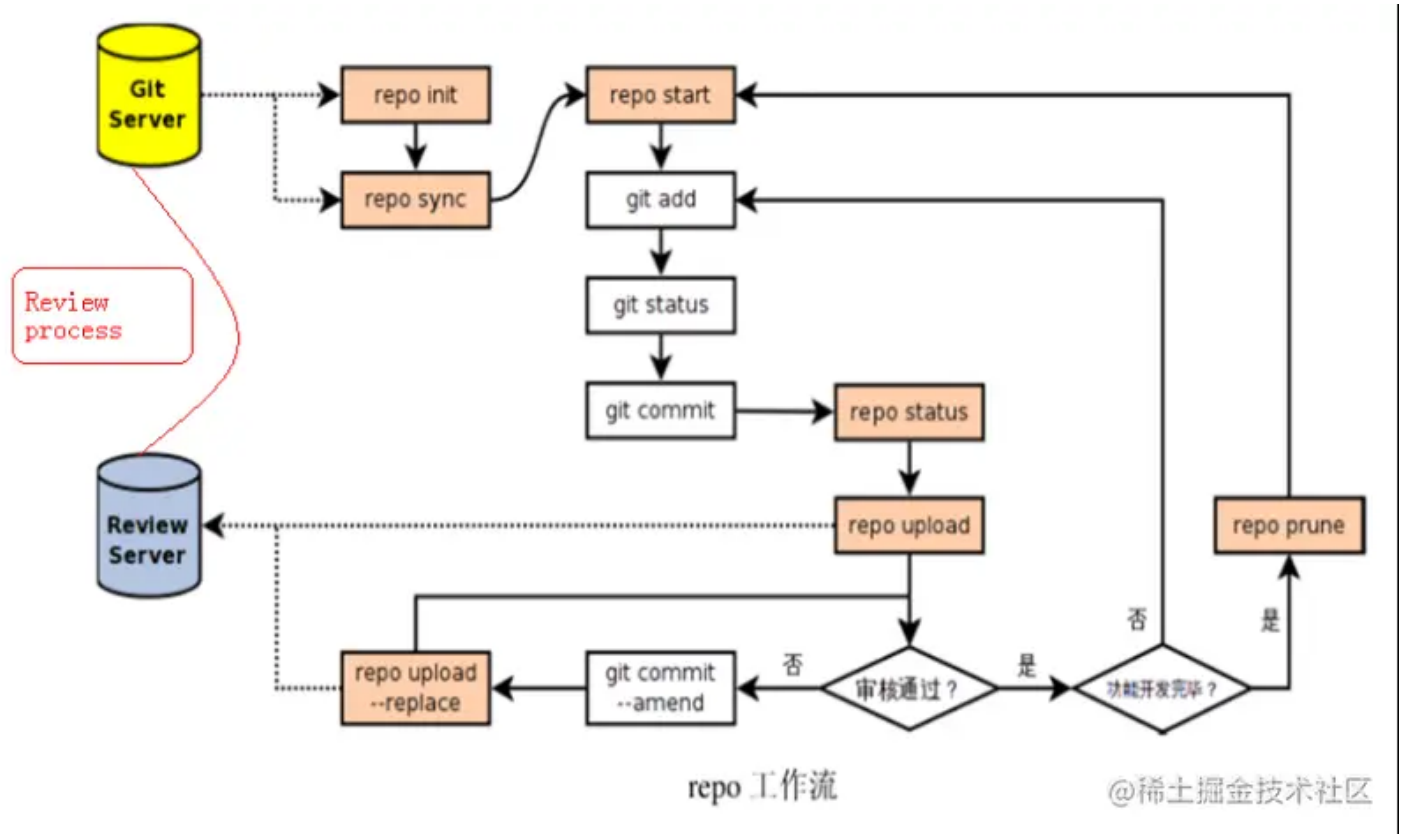
可能遇到的问题

- repo sync导致rebase失败：
    - 失败原因都是本地有修改。

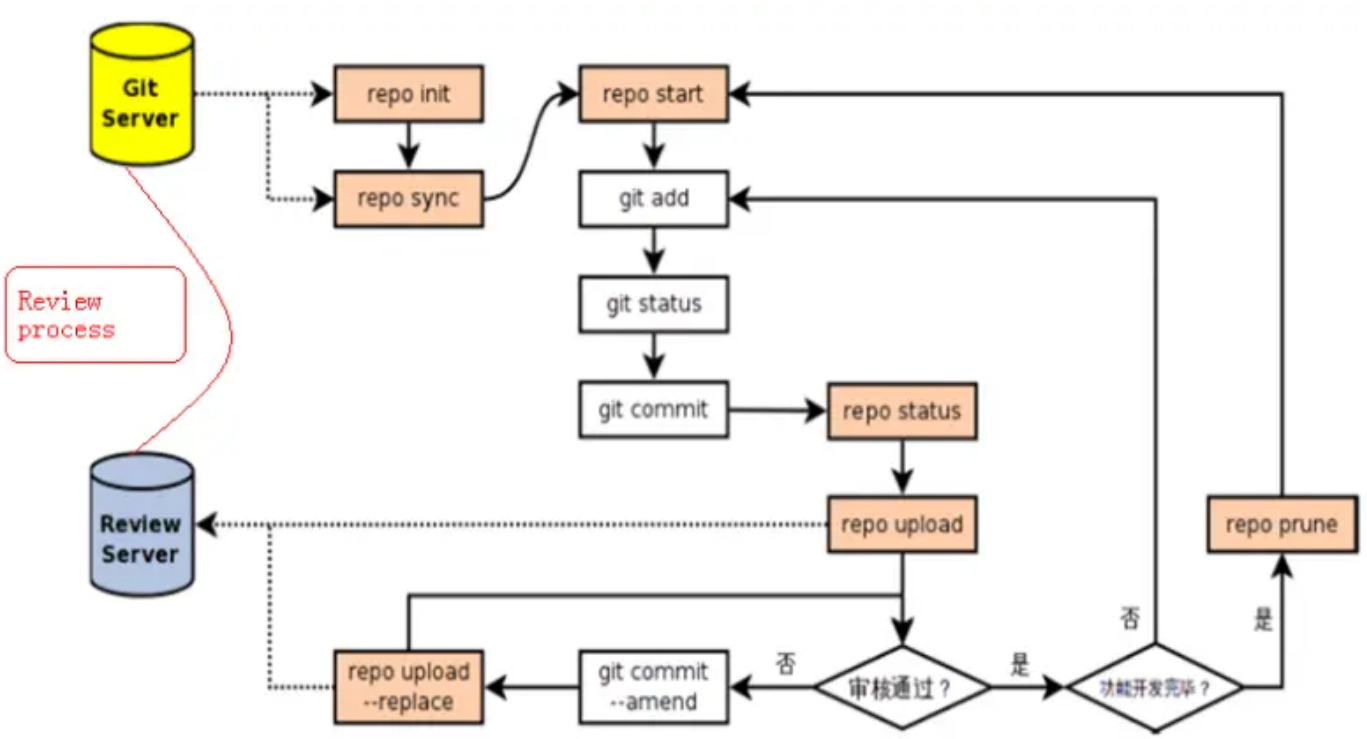
- 方法1：在新的分支上repo sync $cd * failed - error - git - dir$*git rebase –abort $gitcheckoutm/master$repo sync
- 方法2： 强制删除缓存的git库 $rmpackages/apps/Settings–rf$rm .repo/projects/packages/apps/Settings.git –rf $repo sync

最后附图一张：

Google眼中的repo 工作流：



即每个rom的项目由：平台基线部分、rom框架部分，公共库三部分构成。



repo 工作流