链滴

Git Submodule 管理项目子模块

作者：zwxbest

原文链接：https://ld246.com/article/1541567150043

来源网站：链滴

<h2 id="使用场景">使用场景</h2>
当项目越来越庞大之后，不可避免的要拆分成多个子模块，我们希望各个子模块有独立的版本管，并且由专门的人去维护，这时候我们就要用到 git 的 submodule 功能。
<h2 id="常用命令">常用命令</h2>
<pre><code class="highlight-chroma">git clone <repository> --recursive 递归的方式克隆整个项目
git submodule add <repository> <path> 添加子模块
git submodule init 初始化子模块
git submodule update 更新子模块
git submodule foreach git pull 拉取所有子模块
</code></pre>
<h2 id="如何使用">如何使用</h2>
<h3 id="1-创建带子模块的版本库">1. 创建带子模块的版本库</h3>
例如我们要创建如下结构的项目
<pre><code class="highlight-chroma">project
  |--moduleA
  |--readme.txt
</code></pre>
创建 project 版本库，并提交 readme.txt 文件
<pre><code class="highlight-chroma">git init --bare project.git
git clone project.git project1
cd project1
echo "This is a project." > readme.txt
git add .
git commit -m "add readme.txt"
git push origin master
cd ..
</code></pre>
创建 moduleA 版本库，并提交 a.txt 文件
<pre><code class="highlight-chroma">git init --bare moduleA.git
git clone moduleA.git moduleA1
cd moduleA1
echo "This is a submodule." > a.txt
git add .
git commit -m "add a.txt"
git push origin master
cd ..
</code></pre>
在 project 项目中引入子模块 moduleA，并提交子模块信息

<pre><code class="highlight-chroma"><span class="highlight
cl">cd project1
git submodule ad
../moduleA.git moduleA
git status
git diff
git add .
git commit -m "a
d submodule"
git push origin ma
ter
cd ..
</code></pre>
使用 <code>git status</code> 可以看到多了两个需要提交的文件，其中 <code>.gitmodule
</code> 指定 submodule 的主要信息，包括子模块的路径和地址信息，<code>moduleA</code>
指定了子模块的 commit id，使用 <code>git diff</code> 可以看到这两项的内容。这里需要指出
项目的 git 并不会记录 submodule 的文件变动，它是按照 commit id 指定 submodule 的 git head
r，所以 <code>.gitmodules</code> 和 <code>moduleA</code> 这两项是需要提交到父项目
远程仓库的。
<pre><code class="highlight-chroma"><span class="highlight
cl">On branch master
Your branch is up-
o-date with 'origin/master'.
Changes to be c
mmitted:
  (use "git reset H
AD <file>..." to unstage)
    new file: .gitm
dules
    new file: modu
eA
</code></pre>
<h3 id="2-克隆带子模块的版本库">2. 克隆带子模块的版本库</h3>
方法一，先 clone 父项目，再初始化 submodule，最后更新 submodule，初始化只需要做一
，之后每次只需要直接 update 就可以了，需要注意 submodule 默认是不在任何分支上的，它指向
项目存储的 submodule commit id。
<pre><code class="highlight-chroma"><span class="highlight
cl">git clone project.git project2
cd project2
git submodule init
git submodule up
ate
cd ..
</code></pre>
方法二，采用递归参数 <code>--recursive</code>，需要注意同样 submodule 默认是不在
何分支上的，它指向父项目存储的 submodule commit id。
<pre><code class="highlight-chroma"><span class="highlight
cl">git clone project.git project3 --recursive
</code></pre>
<h3 id="3-修改子模块">3. 修改子模块</h3>
修改子模块之后只对子模块的版本库产生影响，对父项目的版本库不会产生任何影响，如果父项
需要用到最新的子模块代码，我们需要更新父项目中 submodule commit id，默认的我们使用 <cod
>git status</code> 就可以看到父项目中 submodule commit id 已经改变了，我们只需要再次提
就可以了。

<pre><code class="highlight-chroma">cd project1/moduleA
git branch
echo "This is a su module." > b.txt
git add .
git commit -m "a d b.txt"
git push origin ma ter
cd ..
git status
git diff
git add .
git commit -m "u date submodule add b.txt"
git push origin ma ter
cd ..
</code></pre>
<h3 id="4-更新子模块">4. 更新子模块</h3>
更新子模块的时候要注意子模块的分支默认不是 master。
方法一，先 pull 父项目，然后执行 <code>git submodule update</code>，注意 moduleA 的分支始终不是 master。
<pre><code class="highlight-chroma">cd project2
git pull
git submodule up ate
cd ..
</code></pre>
方法二，先进入子模块，然后切换到需要的分支，这里是 master 分支，然后对子模块 pull，这方法会改变子模块的分支。
<pre><code class="highlight-chroma">cd project3/moduleA
git checkout mast r
cd ..
git submodule for ach git pull
cd ..
</code></pre>
<h3 id="更换子模块的远程版本库">更换子模块的远程版本库</h3>
比如我想更换为我 fork 过来的远程版本库

编辑.gitmodules
git submodule update --init --recursive --remote

<h3 id="5-删除子模块">5. 删除子模块</h3>
网上有好多用的是下面这种方法
<pre><code class="highlight-chroma">git rm --cached moduleA
rm -rf moduleA
rm .gitmodules

vim .git/config
</code></pre>
删除 submodule 相关的内容，例如下面的内容
<pre><code class="highlight-chroma"><span class="highlight
cl">[submodule "moduleA"]
	url = /Users/ni
k/dev/nick-doc/testGitSubmodule/moduleA.git
</code></pre>
然后提交到远程服务器
<pre><code class="highlight-chroma"><span class="highlight
cl">git add .
git commit -m "r
move submodule"
</code></pre>
但是我自己本地实验的时候，发现用下面的方式也可以，服务器记录的是 <code>.gitmodules<
code> 和 <code>moduleA</code>，本地只要用 git 的删除命令删除 moduleA，再用 git status
查看状态就会发现.gitmodules 和 moduleA 这两项都已经改变了，至于.git/config，仍会记录 sub
odule 信息，但是本地使用也没发现有什么影响，如果重新从服务器克隆则.git/config 中不会有 sub
odule 信息。
<pre><code class="highlight-chroma"><span class="highlight
cl">git rm moduleA
git status
git commit -m "r
move submodule"
git push origin ma
ter
</code></pre>