# Git-submodule使用

说明：Git Submodule 中文通常翻译为 Git 子模块，它允许我们在一个代码仓库中优雅的嵌套使用另外一个或多个代码仓库，从而达到减少代码重复提交，优化工作流程的目的，是内部团队和开源社区协作都非常依赖的 Git 特性。

**一、创建子模块**

在主代码仓库里面创建子模块引用，执行以下指令（以hyy仓库为例）即可：

# Usage

$ git submodule add [url] [path]

# With path

$ git submodule add http://git.tech.nanhaicorp.com.cn/oristarcloud/ctm.git src/ctm

# Without path

$ cd src

$ git submodule add http://git.tech.nanhaicorp.com.cn/oristarcloud/ctm.git

**二、子模块引用文件**

在 hyy 这个仓库下面多出了一个 .gitmodules 文件，该文件列出了所包含的子模块列表，并为列表中每一个子模块指定了本地路径（path）和远程仓库地址（url），除此以外我们还可选为子模块指定 branch 分支，不指定默认为 master 分支。

[submodule "src/ctm"]

path = src/ctm

url = http://git.tech.nanhaicorp.com.cn/oristarcloud/ctm.git

**三、查看子模块**

要查看当前代码仓库所使用的子模块及其状态，除了看 .gitmodules 文件外，还可以执行 git submodule 命令。

$ git submodule

# 已检出子模块代码

5d8101fe20220a22253a73b8593da2682fe7a6d4 src/ccm (heads/master)

# 前面带 - 表示未检出代码，子模块是空文件夹

-eefa3ba35e8b4e8b8bad415bbc50aec5acc9cfab src/rpt (heads/master)

**四、克隆包含子模块的项目**

当你需要克隆一个包含了子模块的远程代码仓库，有两种方式（第二种更加简洁，更优）。

# Clone => Init => Update

$ git clone http://git.tech.nanhaicorp.com.cn/oristarcloud/hyy.git

$ git submodule init

$ git submodule update

# Clone recursive

$ git clone --recursive http://git.tech.nanhaicorp.com.cn/oristarcloud/hyy.git

**五、拉取子模块更新**

拉取子模块更新不再需要 clone 和 init 操作，只需 update 即可，当你的主代码仓库执行 pull 或者切换分支操作后，别忘了执行 update 操作，以保证子模块中的代码与新的 .gitmodules 中版本一致。为了防止误提交旧的子模块依赖信息，每次执行 pull 后，可使用 git status 查看文件状态。

# After Pull

$ git pull http://git.tech.nanhaicorp.com.cn/oristarcloud/hyy.git

# After Checkout

$ git checkout -b develop origin/develop

# You need

$ git status -s

$ git submodule update

# 在主库目录下，将所有子库从上游拉取数据并合并到本地

$ git submodule update --remote --merge

**六、提交子模块修改**

当你需要对当前使用的某个子模块进行修改，并且希望所做修改能够提交到子模块的主仓库，一定要记得切换到 master 分支再修改并提交。

$ cd src/ctm

$ git checkout master

$ git add .

$ git commit -m "Create shortcode for stackblitz"

$ git push orgin master

**七、删除子模块**

Git 中删除子模块略微麻烦一些，因为目前还没有 git submodule rm 这样的命令行，我们要做很多工作才能删得干净：

$ git submodule deinit src/ctm

$ vim .gitmodules # 移除要删除的子模块

$ git add .gitmodules

$ git rm --cached src/ctm

$ rm -rf .git/modules/src/ctm

$ rm -rf src/ctm

$ git commit -m "Remove submodule src/ctm"

**八、子模块批处理**

对于像 src/ 这种包含了多个子模块的仓库怎么管理呢，使用 foreach 循环指令就可以啦。

# Checkout master then pull

$ git submodule foreach 'git checkout master; git pull'

**git submodule使用的注意事项：**

1.有些时候你需要对 submodule 做一些修改，很常见的做法就是切到 submodule 的目录，然后做修改，然后 commit 和 push。  
这里的坑在于，默认 git submodule update 并不会将 submodule 切到任何 branch，所以，默认下 submodule 的 HEAD 是处于游离状态的 (‘detached HEAD’ state)。所以在修改前，记得一定要用 git checkout master 将当前的 submodule 分支切换到 master，然后才能做修改和提交。

2.如果你不慎忘记切换到 master 分支，又做了提交，可以用 cherry-pick 命令挽救。具体做法如下：  
用 git checkout master 将 HEAD 从游离状态切换到 master 分支 , 这时候，git 会报 Warning 说有一个提交没有在 branch 上，记住这个提交的 change-id（假如 change-id 为 aaaa)  
用 git cherry-pick aaaa 来将刚刚的提交作用在 master 分支上  
用 git push 将更新提交到远程版本库中

3.git pull 之后，立即执行 git status, 如果发现 submodule 有修改，立即执行 git submodule update

4.尽量不要使用 git commit -a， git add 命令存在的意义就是让你对加入暂存区的文件做二次确认，而 git commit -a 相当于跳过了这个确认过程。

5.子库开发者只开发提交各自子库，对于主库只做拉取操作，主库和子库、子库和子库之间，主要各自的开发分支，联调时注意切换到对应分支上，一定要注意默认 git submodule update 并不会将 submodule 切到任何 branch

参考资料：

1.git官方对于submodule的使用： [Git 工具 - 子模块](https://git-scm.com/book/zh/v2/Git-工具-子模块)