

《GIT应用之子模块》

通常我们在工作过程中，需要在一个工程中嵌套另一个工程——也许是一个依赖的第三方库或者是协同开发者开发的工程的另一部分。如此就诞生了这样的需求：你需要在一个仓库中工作，处理其他仓库且又需要保持它们之间相互独立。

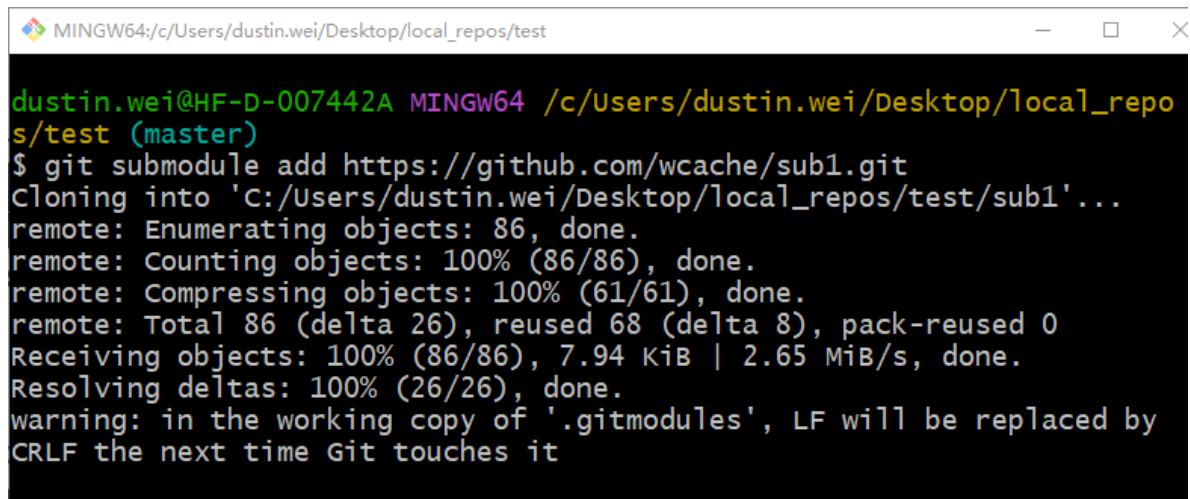
Git 使用一种叫做 **submodules** 的特性，来满足上述需求。

submodules 允许我们在一个仓库中嵌套另一个仓库(子目录形式存在)。这样就可以在你的仓库中 clone 其他仓库，且保持提交独立。

添加子模块

接下来，让我们在项目仓库 `test` 目录中，添加子仓库 `https://github.com/wcache/sub1.git`。

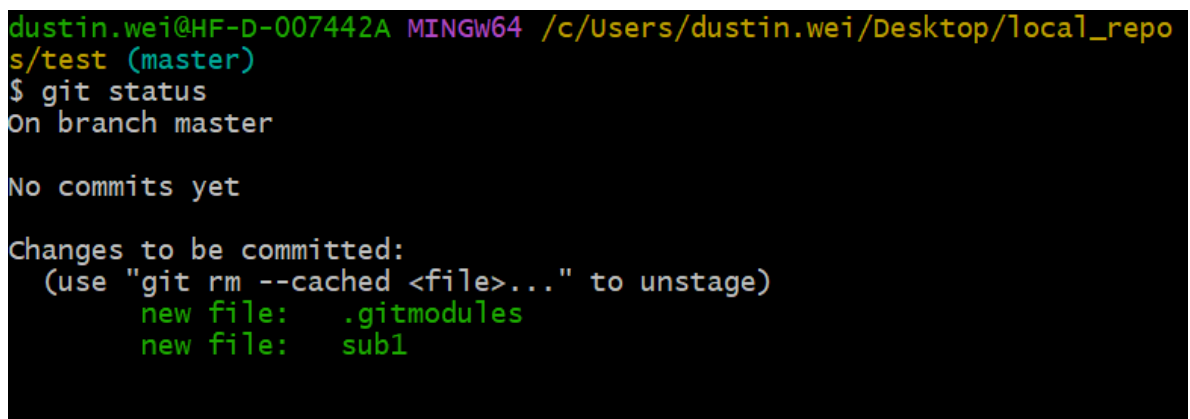
命令：`git submodule add https://github.com/wcache/sub1.git`

A terminal window titled 'MINGW64:/c/Users/dustin.wei/Desktop/local_repos/test' showing the execution of the command 'git submodule add https://github.com/wcache/sub1.git'. The output shows the cloning process, including enumerating and counting objects, compressing them, and resolving deltas. A warning message indicates that LF line endings will be replaced by CRLF in the working copy of '.gitmodules' the next time Git touches it.

```
dustin.wei@HF-D-007442A MINGW64 /c/Users/dustin.wei/Desktop/local_repos/test (master)
$ git submodule add https://github.com/wcache/sub1.git
Cloning into 'C:/Users/dustin.wei/Desktop/local_repos/test/sub1'...
remote: Enumerating objects: 86, done.
remote: Counting objects: 100% (86/86), done.
remote: Compressing objects: 100% (61/61), done.
remote: Total 86 (delta 26), reused 68 (delta 8), pack-reused 0
Receiving objects: 100% (86/86), 7.94 KiB | 2.65 MiB/s, done.
Resolving deltas: 100% (26/26), done.
warning: in the working copy of '.gitmodules', LF will be replaced by
CRLF the next time Git touches it
```

默认情况下，子模块将会以与子仓库同名目录形式存在。可以在子仓库 url 后面添加路径来指定子目录。

使用 `git status` 查看仓库当前状态。

A terminal window showing the output of the 'git status' command. It indicates that there are no commits yet and lists the changes to be committed: a new file named '.gitmodules' and a new file named 'sub1'.

```
dustin.wei@HF-D-007442A MINGW64 /c/Users/dustin.wei/Desktop/local_repos/test (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   .gitmodules
        new file:   sub1
```

可以看到，我们多个 `sub1` 文件夹和 `.gitmodules` 隐藏文件。

其中，`sub1` 是我们子仓库存放的目录。`.gitmodules` 是当前主仓库用于记录跟踪子仓库的文件，如下图：

```
dustin.wei@HF-D-007442A MINGW64 /c/Users/dustin.wei/Desktop/local_repos/test (master)
$ cat .gitmodules
[submodule "sub1"]
    path = sub1
    url = https://github.com/wcache/sub1.git
```

标签[submodule "sub1"]表明一个跟踪的子仓库，path为存储路径，url为子仓库地址。

注意，此时我们仅仅在本地仓库添加了一个子仓库，在主仓库没有新提交之前，其他的协同开发者，是无法获取子仓库的。

我们需要将添加子仓库后的变更，在主仓库中新提交。

```
git add .
git commit -m 'add sub1'
git push origin master
```

克隆一个带有子模块的仓库

```
dustin.wei@HF-D-007442A MINGW64 /c/Users/dustin.wei/Desktop/local_repos
$ git clone file:///c:/users/dustin.wei/desktop/local_repos/test.git
Cloning into 'test'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (3/3), done.
```

当我们另一位开发者，clone仓库之后，发现子仓库目录sub1为空。

```
dustin.wei@HF-D-007442A MINGW64 /c/Users/dustin.wei/Desktop/local_repos/test (master)
$ ls
sub1/

dustin.wei@HF-D-007442A MINGW64 /c/Users/dustin.wei/Desktop/local_repos/test (master)
$ ls sub1/
```

这个时候，我们需要2个步骤来初始化子仓库。

- `git submodule init`
- `git submodule update`

```
dustin.wei@HF-D-007442A MINGW64 /c/Users/dustin.wei/Desktop/local_repos/test (master)
$ git submodule init
Submodule 'sub1' (https://github.com/wcache/sub1.git) registered for path 'sub1'

dustin.wei@HF-D-007442A MINGW64 /c/Users/dustin.wei/Desktop/local_repos/test (master)
$ git submodule update
Cloning into 'C:/Users/dustin.wei/Desktop/local_repos/test/sub1'...
Submodule path 'sub1': checked out 'b704c88557b5766a7ae13bd449e2012f0504c08a'

dustin.wei@HF-D-007442A MINGW64 /c/Users/dustin.wei/Desktop/local_repos/test (master)
$ ls sub1/
README.md  components/
```