Git as I understand (8): Subtree and Submodule

1. Module in Software

"Divide and conquer", "Separation of concerns" are heavily used strategy utilized in software design and implementation.Under such design philosophy, We might have well defined class interface, or java/c++ package which provides clear-cut functionality, a binary jar/share library(so), or even a sub-system which can run on its own. Per my understanding, any software artifacts which can provide a well defined and relatively stable interface and provide services for other software should ultimately be re-factored into a module in different level (class/package/share library/standalone system).

Accordingly, we have the requirement to manage modules during software requirement. For version control perspective, if part of software have stable API interface, could be delivered in its own release plan, then there is reason to manage it in a separate module. Modulation brings benefits like:

a. Less integration build failure
If code change of all modules of different maturation level are propagated into integrate branch without any control, developer might face daily build failure brought about by other modules. It is a productivity killer; With module, we can avoid a error-prone integration and release window;

b. Less source code in one sub-project
If modules is loosely coupled, we might use the built binary of other modules instead of introducing a bunch of source code that we are not responsible for

c. Third party project
We might use third party project which is out of our control. We should have module support for such project

2. Git and Linux Kernel

Git was first designed to support Linux Kernel development. Linux Kernel is a huge project in itself, and Git maintains it under one repository. Via the help of branch, we can avoid the interference of various parallel development activities. With the help of distributed repositories and "lieutenant/dictator" work flow, Linux Kernel can manage its integration and release cycle very well. Linux Kernel is itself very modular. With the help of its make file build system, developer could build their modules easily without the headache of maintaining convoluted make files. The source code base of Linux Kernel is huge. But with help of tools mentioned above (branch, distributed repository and highly efficient distributed workflow, and more important, highly skilled kernel developers), Linux Kernel has evolved with reputation of stable quality and well-organized release cycles.

3. Requirement of sub-project management

Git works so well in managing Linux Kernel via one project/one repository, why some people have the the concept of "sub-project" at any rate? Why not just adopt Linux Kernel's model? Do they have project which is more complex than Linux Kernel?

Most likely not. The reason to raise the request for "sub-project" in Git, per my understanding, resides in:

a. organization model of development activity

No single organization is fully responsible for Linux Kernel. There are more than 200 organization and more individual developers who actively involve in development and maintenance of Linux Kernel. With so many stakeholders, Linux Kernel is still a single project with well defined unified release schedule. It is fair to say Linux Kernel is developed by one huge "virtual organization".

For our humble project, the story is different. Nowadays, no one would like to create a new wheel if he can find support from open source projects. We won't create our own xml parser, font rendering library, work flow engine, etc, etc. When we employed third party module into our small humble project, We naturally have the needs to manage a separate "sub-project" since:

. Third party project has its own organization of development activity and release cycle. It is not necessary or always possible to synchronize its development/release with our project;
. Third party module has clear interface, and we are not responsible of its development. We just use them. Many times, a stable release of such project suffices to support our project.

Even within an software organization which will deliver a suite product, although there is a unified release schedule for the whole suite, there might be different release cycle for sub-system or sub modules. It is a time and complexity saver to have different sub-projects under one big umbrella in version control system. Such organization will reduce the risk of integration hazard and increase overall product delivery, since we can have separate control over sub-system/module with various naturalization levels.

b. human resource and skill set

Even with the requirement mentioned in item a, some might still argue the Linux Kernel model still works. In Linux Kernel, sub-systems are handled by various level of "gatekeeper". They are responsible for what should be integrated into Kernel mainline via delicate work flow in a distributed manner.

Linux Virtual Console(2)…
Mac book pro 11.3 Linux…
How to debug system p…

Linux Virtual Console (1…
Mac book pro 11.3 Linux…
How to Bootstrap differe…

How to set up Sof… 2
Mac book pro 11.3 Linux…
DELL R720 iDRAC7 fir…

GRUB2 How To (7): De…
Mac book pro 11.3 Linux…
DELL R710 iDRAC6 set…

GRUB2 How To (6): PX…
Mac book pro 11.3 Linux…
The best way to configur…

GRUB2 How To (5): Buil…
Linux From Scratch for …
Ec2 linux server s… 1

GRUB2 How To (… 2
Linux From Scratch for …
Linux Virtual Console(6)…

GRUB2 How To (3) : UE…
Linux From Scratch for …
dnscrypt proxy in Archli…

GRUB2 How To (… 4
Linux From Scratch for …
Git as I understand (9): …

Debugging with QEMU …
Linux From Scratch for …
Linux Virtual Console(5)…

GRUB2 How To (1) : Ma…

Git as I understand (8): …

QEMU 1.5 build configur…
Linux From Scratch for …
Git as I understand (6): …

Linux From Scratch for …
Git as I understand (7): …

Linux From Scratch for …
Git as I understand (5): …

Linux From Scratch for …
Git as I understand (4): …

Linux From Scratch for …
Git as I understand (3): …

Linux From Scratch for …
Git as I understand (2): r…

Linux Virtual Console(4)…
Git as I understand (1): …

Linux Virtual Console(3)…
Mac book pro 11.3 Linux…

Regular Expression: A …
Mac book pro 11.3 Linux…

Automatic Scripts… 2
Mac book pro 11.3 Linux…

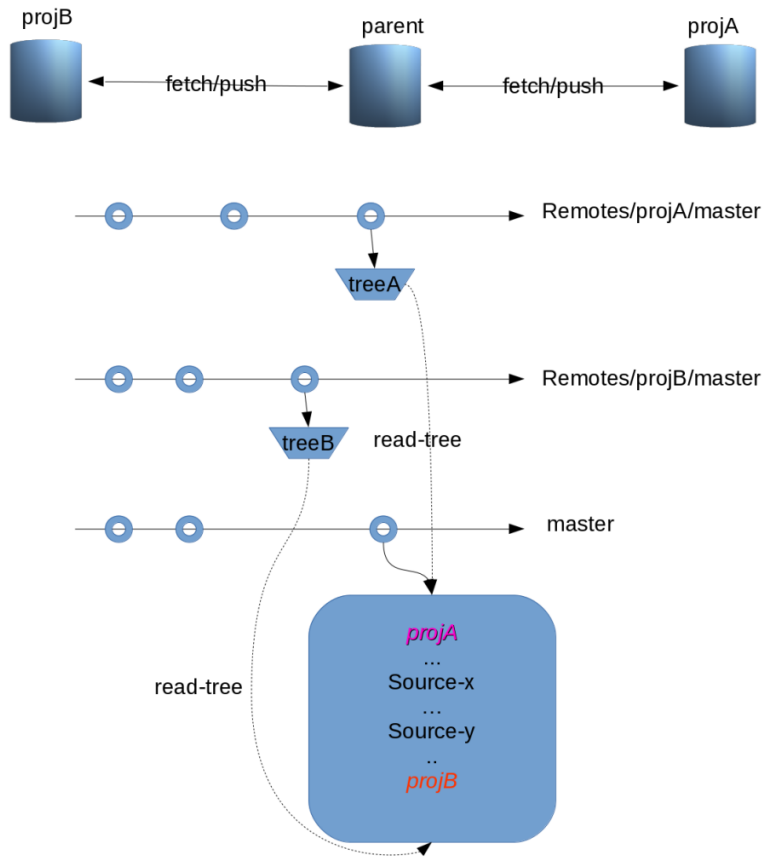
All in all, because of impact of a and b, many organizations/individuals will resort back to traditional project/sub-project model to handle complexity and alleviate management burden.

4. subtree project support in Git

Per my understanding, sub-project support in Git is an "afterthought". It is a high level "patch" for software organizations who are accustomed to traditional "sub-project" work flow. Sub-projects often appears as a sub-directory under parent project. There is no difference under Git for this convention.

a. One repository, different branch, read-tree/update-index

In this case, different sub projects resides in different branch. In local development branch, we use "read-index/update-index" to implant sub-project root tree object into a sub-directory of dev branch's root tree. Every time sub-project upstream has updates, we implant another version of the sub-project root tree into our dev branch' root tree.



[http://2.bp.blogspot.com/-0v7tBnrb7cE/U6KE-gSAWzI/AAAAAAAABrc/Of-

Linux Virtual Console(2)…
Mac book pro 11.3 Linux…
How to debug system p…

Linux Virtual Console (1…
Mac book pro 11.3 Linux…
How to Bootstrap differe…

How to set up Sof… 2
Mac book pro 11.3 Linux…
DELL R720 iDRAC7 fir…


GRUB2 How To (7): De…
Mac book pro 11.3 Linux…
DELL R710 iDRAC6 set…


GRUB2 How To (6): PX…
Mac book pro 11.3 Linux…
The best way to configur…


GRUB2 How To (5): Buil…
Linux From Scratch for …
Ec2 linux server s… 1


GRUB2 How To (… 2
Linux From Scratch for …
Linux Virtual Console(6)…


GRUB2 How To (3) : UE…
Linux From Scratch for …
dnscrypt proxy in Archli…


GRUB2 How To (… 4
Linux From Scratch for …
Git as I understand (9): …


Debugging with QEMU …
Linux From Scratch for …
Linux Virtual Console(5)…

GRUB2 How To (1) : Ma…


Git as I understand (8): …


QEMU 1.5 build configur…
Linux From Scratch for …
Git as I understand (6): …


Linux From Scratch for …
Git as I understand (7): …


Linux From Scratch for …
Git as I understand (5): …


Linux From Scratch for …
Git as I understand (4): …


Linux From Scratch for …
Git as I understand (3): …

Linux From Scratch for …
Git as I understand (2): r…


Linux Virtual Console(4)…
Git as I understand (1): …

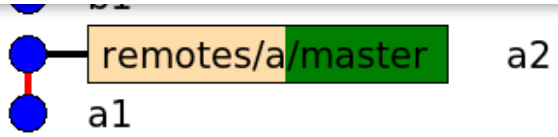
Linux Virtual Console(3)…
Mac book pro 11.3 Linux…


Regular Expression: A …
Mac book pro 11.3 Linux…


Automatic Scripts… 2
Mac book pro 11.3 Linux

```
[luke@rmbp git]$ git init parent
Initialized empty Git repository in /tmp/git/parent/.git/
[luke@rmbp git]$ git init projA
Initialized empty Git repository in /tmp/git/projA/.git/
[luke@rmbp git]$ git init projB
Initialized empty Git repository in /tmp/git/projB/.git/
[luke@rmbp git]$ cd projA/
[luke@rmbp projA]$ git simple-commit a1 a2
[master (root-commit) f08acc1] a1
 1 file changed, 1 insertion(+)
 create mode 100644 a1.txt
[master ca28b1c] a2
 1 file changed, 1 insertion(+)
 create mode 100644 a2.txt
[luke@rmbp projA]$ cd ../projB
[luke@rmbp projB]$ git simple-commit b1 b2 3
[master (root-commit) 8383efc] b1
 1 file changed, 1 insertion(+)
 create mode 100644 b1.txt
[master 4458f31] b2
 1 file changed, 1 insertion(+)
 create mode 100644 b2.txt
[master 8c012ef] 3
 1 file changed, 1 insertion(+)
 create mode 100644 3.txt
[luke@rmbp projB]$ cd ../parent/

[luke@rmbp parent]$ git remote add a /tmp/git/projA
[luke@rmbp parent]$ git remote add b /tmp/git/projB
[luke@rmbp parent]$ git fetch a
remote: Counting objects: 6, done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 6 (delta 0), reused 0 (delta 0)
Unpacking objects: 100% (6/6), done.
From /tmp/git/projA
 * [new branch]      master     -> a/master
[luke@rmbp parent]$ git fetch b
warning: no common commits
remote: Counting objects: 9, done.
remote: Compressing objects: 100% (5/5), done.
remote: Total 9 (delta 0), reused 0 (delta 0)
Unpacking objects: 100% (9/9), done.
From /tmp/git/projB
 * [new branch]      master     -> b/master
[luke@rmbp parent]$ gitk --all
```



[http://3.bp.blogspot.com/-DtAbwiiMh9g/U6KGgLbvXTI/AAAAAAAABro/Th6FOtGwk-4/s1600/gitk17.png]

Here we can see we have two disparate remote tracking branches. They don't even share a common commit. Now let us add some commits in our local master branch:

```
[master (root-commit) 1f77b3f] m1
 1 file changed, 1 insertion(+)
 create mode 100644 m1.txt
[master 0ebbc4a] m2
 1 file changed, 1 insertion(+)
 create mode 100644 m2.txt
[luke@rmbp parent]$ git l
0ebbc4a m2
1f77b3f m1
[luke@rmbp parent]$ git cat-file -p master
tree c2698574830f59f48b76cf039506cee89e74e3fe
parent 1f77b3f692cd2837f801dddaa9f3d1c4e8922501
author Luke Luo <luke.jf.luo@gmail.com> 1403160313 +0800
committer Luke Luo <luke.jf.luo@gmail.com> 1403160313 +0800

m2
[luke@rmbp parent]$ git ls-tree c2698574830f59f48b76cf039506cee89e74e3fe
100644 blob 63a911f26fe84ea7fd8a863a636cfac908895ec9 m1.txt
100644 blob 08bb2331e777f431177c40df6841c0034f89fb58 m2.txt
[luke@rmbp parent]$ git ls-files -s
100644 63a911f26fe84ea7fd8a863a636cfac908895ec9 0 m1.txt
100644 08bb2331e777f431177c40df6841c0034f89fb58 0 m2.txt
```

Here we can see the master HEAD's root tree have "m1.txt/m2.txt". Now let add sub-projects' root tree into master's tree:

```
[luke@rmbp parent]$ git cat-file -p remotes/a/master
tree 09efc7e59a839528ac7bda9fa020dc9101278680
parent f08acc1d1093999bd203a0efb70d10d492bb409b
author Luke Luo <luke.jf.luo@gmail.com> 1403159999 +0800
committer Luke Luo <luke.jf.luo@gmail.com> 1403159999 +0800

a2
[luke@rmbp parent]$ git cat-file -p remotes/b/master
tree d6e6a288ee17ef844941e21d3f94ce6a79f8bfb0
parent 4458f31a5dff2bd32dfeb86b491d26edca48b49a
author Luke Luo <luke.jf.luo@gmail.com> 1403160011 +0800
```

Linux Virtual Console(2)…
Mac book pro 11.3 Linux…
How to debug system p…

Linux Virtual Console (1…
Mac book pro 11.3 Linux…
How to Bootstrap differe…

How to set up Sof… 2
Mac book pro 11.3 Linux…
DELL R720 iDRAC7 fir…

GRUB2 How To (7): De…
Mac book pro 11.3 Linux…
DELL R710 iDRAC6 set…

GRUB2 How To (6): PX…
Mac book pro 11.3 Linux…
The best way to configur…

GRUB2 How To (5): Buil…
Linux From Scratch for …
Ec2 linux server s… 1

GRUB2 How To (… 2
Linux From Scratch for …
Linux Virtual Console(6)…

GRUB2 How To (3) : UE…
Linux From Scratch for …
dnscrypt proxy in Archli…

GRUB2 How To (… 4
Linux From Scratch for …
Git as I understand (9): …

Debugging with QEMU …
Linux From Scratch for …
Linux Virtual Console(5)…

GRUB2 How To (1) : Ma…

Git as I understand (8): …

QEMU 1.5 build configur…
Linux From Scratch for …
Git as I understand (6): …

Linux From Scratch for …
Git as I understand (7): …

Linux From Scratch for …
Git as I understand (5): …

Linux From Scratch for …
Git as I understand (4): …

Linux From Scratch for …
Git as I understand (3): …

Linux From Scratch for …
Git as I understand (2): r…

Linux Virtual Console(4)…
Git as I understand (1): …

Linux Virtual Console(3)…
Mac book pro 11.3 Linux…

Regular Expression: A …
Mac book pro 11.3 Linux…

Automatic Scripts… 2
Mac book pro 11.3 Linux…

```
100644 08bb2331e777f431177c40df6841c0034f89fb58 0 m2.txt
[luke@rmbp parent]$ git ls-tree master
100644 blob 63a911f26fe84ea7fd8a863a636cfac908895ec9 m1.txt
100644 blob 08bb2331e777f431177c40df6841c0034f89fb58 m2.txt
[luke@rmbp parent]$ git read-tree --prefix=projA/ remotes/a/master^{tree}
[luke@rmbp parent]$ git ls-files -s
100644 63a911f26fe84ea7fd8a863a636cfac908895ec9 0 m1.txt
100644 08bb2331e777f431177c40df6841c0034f89fb58 0 m2.txt
100644 da0f8ed91a8f2f0f067b3bdf26265d5ca48cf82c 0 projA/a1.txt
100644 c1827f07e114c20547dc6a7296588870a4b5b62c 0 projA/a2.txt
[luke@rmbp parent]$ git read-tree --prefix=projB/ remotes/b/master^{tree}
[luke@rmbp parent]$ git ls-files -s
100644 63a911f26fe84ea7fd8a863a636cfac908895ec9 0 m1.txt
100644 08bb2331e777f431177c40df6841c0034f89fb58 0 m2.txt
100644 da0f8ed91a8f2f0f067b3bdf26265d5ca48cf82c 0 projA/a1.txt
100644 c1827f07e114c20547dc6a7296588870a4b5b62c 0 projA/a2.txt
100644 00750edc07d6415dcc07ae0351e9397b0222b7ba 0 projB/3.txt
100644 c9c6af7f78bc47490dbf3e822cf2f3c24d4b9061 0 projB/b1.txt
100644 e6bfff5c1d0f0ecd501552b43a1e13d8008abc31 0 projB/b2.txt
[luke@rmbp parent]$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

 new file:   projA/a1.txt
 new file:   projA/a2.txt
 new file:   projB/3.txt
 new file:   projB/b1.txt
 new file:   projB/b2.txt

Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

 deleted:    projA/a1.txt
 deleted:    projA/a2.txt
 deleted:    projB/3.txt
 deleted:    projB/b1.txt
 deleted:    projB/b2.txt
[luke@rmbp parent]$ git checkout-index -a
[luke@rmbp parent]$ tree
.
├── m1.txt
├── m2.txt
├── projA
│   ├── a1.txt
│   └── a2.txt
```

```
A  projA/a1.txt
A  projA/a2.txt
A  projB/3.txt
A  projB/b1.txt
A  projB/b2.txt



[luke@rmbp parent]$ git commit -m "add sub project A/B tree into master"
On branch master
nothing to commit, working directory clean
[luke@rmbp parent]$ git l
6d64ab3 add sub project A/B tree into master
0ebbc4a m2
1f77b3f m1
```

Now both projA and projB have a snapshot under master HEAD's root tree. Suppose upstream has changes for A/B, we just need to fetch these updates from upstream and re-align sub project tree under master's root dir to current sub project tree version.

```
[luke@rmbp git]$ cd projA
[luke@rmbp projA]$ git simple-commit a3 a4
[master d7d8e7c] a3
 1 file changed, 1 insertion(+)
 create mode 100644 a3.txt
[master 1a53479] a4
 1 file changed, 1 insertion(+)
 create mode 100644 a4.txt
[luke@rmbp projA]$ cd ../projB
[luke@rmbp projB]$ git l
8c012ef 3
4458f31 b2
8383efc b1
[luke@rmbp projB]$ git simple-commit b4 b5
[master 0683a8e] b4
 1 file changed, 1 insertion(+)
 create mode 100644 b4.txt
[master 66ede3b] b5
 1 file changed, 1 insertion(+)
 create mode 100644 b5.txt
[luke@rmbp projB]$ cd ../parent
[luke@rmbp parent]$ git fetch a
remote: Counting objects: 6, done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 6 (delta 1), reused 0 (delta 0)
Unpacking objects: 100% (6/6), done.
From /tmp/git/projA
```

Linux Virtual Console(2)…
Mac book pro 11.3 Linux…
How to debug system p…

Linux Virtual Console (1…
Mac book pro 11.3 Linux…
How to Bootstrap differe…

How to set up Sof… 2
Mac book pro 11.3 Linux…
DELL R720 iDRAC7 fir…

GRUB2 How To (7): De…
Mac book pro 11.3 Linux…
DELL R710 iDRAC6 set…

GRUB2 How To (6): PX…
Mac book pro 11.3 Linux…
The best way to configur…

GRUB2 How To (5): Buil…
Linux From Scratch for …
Ec2 linux server s… 1

GRUB2 How To (… 2
Linux From Scratch for …
Linux Virtual Console(6)…

GRUB2 How To (3) : UE…
Linux From Scratch for …
dnscrypt proxy in Archli…

GRUB2 How To (… 4
Linux From Scratch for …
Git as I understand (9): …

Debugging with QEMU …
Linux From Scratch for …
Linux Virtual Console(5)…

GRUB2 How To (1) : Ma…

Git as I understand (8): …

QEMU 1.5 build configur…
Linux From Scratch for …
Git as I understand (6): …

Linux From Scratch for …
Git as I understand (7): …

Linux From Scratch for …
Git as I understand (5): …

Linux From Scratch for …
Git as I understand (4): …

Linux From Scratch for …
Git as I understand (3): …

Linux From Scratch for …
Git as I understand (2): r…

Linux Virtual Console(4)…
Git as I understand (1): …

Linux Virtual Console(3)…
Mac book pro 11.3 Linux…

Regular Expression: A …
Mac book pro 11.3 Linux…

Automatic Scripts… 2
Mac book pro 11.3 Linux…

```
From /tmp/git/projB
   8c012ef..66ede3b  master     -> b/master
```

Now both projA and projB has new change, let us update our master head's index to reflect this change:

```
[luke@rmbp parent]$ git rm -r --cached projA projB
rm 'projA/a1.txt'
rm 'projA/a2.txt'
rm 'projB/3.txt'
rm 'projB/b1.txt'
rm 'projB/b2.txt'
[luke@rmbp parent]$ git s
## master
D  projA/a1.txt
D  projA/a2.txt
D  projB/3.txt
D  projB/b1.txt
D  projB/b2.txt
?? projA/
?? projB/
[luke@rmbp parent]$ git read-tree --prefix=projA a/master^{tree}
[luke@rmbp parent]$ git read-tree --prefix=projB b/master^{tree}
[luke@rmbp parent]$ git ls-files --stage
100644 63a911f26fe84ea7fd8a863a636cfac908895ec9 0 m1.txt
100644 08bb2331e777f431177c40df6841c0034f89fb58 0 m2.txt
100644 da0f8ed91a8f2f0f067b3bdf26265d5ca48cf82c 0 projA/a1.txt
100644 c1827f07e114c20547dc6a7296588870a4b5b62c 0 projA/a2.txt
100644 d616f7380ad325123fed6f628d02fa76e1ce77c3 0 projA/a3.txt
100644 88ba23dca8c8529f8165539c369925a99391a4d1 0 projA/a4.txt
100644 00750edc07d6415dcc07ae0351e9397b0222b7ba 0 projB/3.txt
100644 c9c6af7f78bc47490dbf3e822cf2f3c24d4b9061 0 projB/b1.txt
100644 e6bfff5c1d0f0ecd501552b43a1e13d8008abc31 0 projB/b2.txt
100644 8e953e84d803f13fd06416a1bd8161dcd93cfd00 0 projB/b4.txt
100644 90a5159bf020296276ea5ca1bcd292a9b1de9947 0 projB/b5.txt
```

First, we remove the existing tree (projA/projB), then we use "git read-tree" to add the new version of trees.

b. subtree merge strategy

In case a, with the help from "read-tree", we already can separate sub-project into remote tracked branch, and we can select which version from upstream and "read-tree" it into our dev branch. In Git merge, with the "subtree" merge strategy and "--squash" option, we can achieve same effect via high level porcelain command. For detailed information on git subtree merge, you can refer to previous post :

```
a
[luke@rmbp parent]$ git remote -v
a /tmp/git/projA/ (fetch)
a /tmp/git/projA/ (push)
[luke@rmbp parent]$ git branch -avv
  remotes/a/master 9f527d9 a2
```

We still need to "read-tree" the remote branch into master branch:

```
[luke@rmbp parent]$ git read-tree -u --prefix=projA a/master^{tree}
[luke@rmbp parent]$ ls
projA
[luke@rmbp parent]$ git status
On branch master


Initial commit


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


 new file:   projA/a1.txt
 new file:   projA/a2.txt


[luke@rmbp parent]$ git commit -m "read-tree sub projA"
[master (root-commit) 9de4739] read-tree sub projA
 2 files changed, 2 insertions(+)
 create mode 100644 projA/a1.txt
 create mode 100644 projA/a2.txt
```

Now we will add more commits in remote projA, then we will fetch the update from parent repository:

```
[luke@rmbp git]$ cd projA/
[luke@rmbp projA]$ git simple-commit a3 a4
[master 1c8922f] a3
 1 file changed, 1 insertion(+)
 create mode 100644 a3.txt
[master 12fd0d0] a4
 1 file changed, 1 insertion(+)
 create mode 100644 a4.txt


[luke@rmbp projA]$ cd ../parent/


[luke@rmbp parent]$ git fetch a


remote: Counting objects: 6, done.


remote: Compressing objects: 100% (4/4), done.
remote: Total 6 (delta 1), reused 0 (delta 0)
Unpacking objects: 100% (6/6), done.
From /tmp/git/projA
```

Linux Virtual Console(2)…
Mac book pro 11.3 Linux…
How to debug system p…

Linux Virtual Console (1…
Mac book pro 11.3 Linux…
How to Bootstrap differe…

How to set up Sof… 2
Mac book pro 11.3 Linux…
DELL R720 iDRAC7 fir…

GRUB2 How To (7): De…
Mac book pro 11.3 Linux…
DELL R710 iDRAC6 set…

GRUB2 How To (6): PX…
Mac book pro 11.3 Linux…
The best way to configur…

GRUB2 How To (5): Buil…
Linux From Scratch for …
Ec2 linux server s… 1

GRUB2 How To (… 2
Linux From Scratch for …
Linux Virtual Console(6)…

GRUB2 How To (3) : UE…
Linux From Scratch for …
dnscrypt proxy in Archli…

GRUB2 How To (… 4
Linux From Scratch for …
Git as I understand (9): …

Debugging with QEMU …
Linux From Scratch for …
Linux Virtual Console(5)…

GRUB2 How To (1) : Ma…

Git as I understand (8): …

QEMU 1.5 build configur…
Linux From Scratch for …
Git as I understand (6): …

Linux From Scratch for …
Git as I understand (7): …

Linux From Scratch for …
Git as I understand (5): …

Linux From Scratch for …
Git as I understand (4): …

Linux From Scratch for …
Git as I understand (3): …

Linux From Scratch for …
Git as I understand (2): r…

Linux Virtual Console(4)…
Git as I understand (1): …

Linux Virtual Console(3)…
Mac book pro 11.3 Linux…

Regular Expression: A …
Mac book pro 11.3 Linux…

Automatic Scripts… 2
Mac book pro 11.3 Linux…

```
.
└── projA
    ├── a1.txt
    └── a2.txt

1 directory, 2 files
[luke@rmbp parent]$ git merge -s recursive -X subtree=projA --no-commit --squash a/master
Squash commit -- not updating HEAD
Automatic merge went well; stopped before committing as requested
[luke@rmbp parent]$ tree
.
└── projA
    ├── a1.txt
    ├── a2.txt
    ├── a3.txt
    └── a4.txt

1 directory, 4 files
[luke@rmbp parent]$ git s
## master
A  projA/a3.txt
A  projA/a4.txt
[luke@rmbp parent]$ git commit -am "merge subproject projA"
[master d5d0456] merge subproject projA
 2 files changed, 2 insertions(+)
 create mode 100644 projA/a3.txt
 create mode 100644 projA/a4.txt

[luke@rmbp parent]$ git l

d5d0456 merge subproject projA

9de4739 read-tree sub projA
```

If we want to make change in master branch under "projA", then push change back to remote repository, we need to make a local tracking branch. We then merge change back to local tracking branch, then we push local tracking branch back to remote repository:

```
[luke@rmbp parent]$ cd projA/
[luke@rmbp projA]$ ls
a1.txt  a2.txt  a3.txt  a4.txt
[luke@rmbp projA]$ echo m1 > m1.txt
[luke@rmbp projA]$ cd ..
[luke@rmbp parent]$ git add projA/
[luke@rmbp parent]$ git commit -m "master change back"
[master 07e59c1] master change back
```

Linux Virtual Console(2)…
Mac book pro 11.3 Linux…
How to debug system p…

Linux Virtual Console (1…
Mac book pro 11.3 Linux…
How to Bootstrap differe…

How to set up Sof… 2
Mac book pro 11.3 Linux…
DELL R720 iDRAC7 fir…

GRUB2 How To (7): De…
Mac book pro 11.3 Linux…
DELL R710 iDRAC6 set…

GRUB2 How To (6): PX…
Mac book pro 11.3 Linux…
The best way to configur…

GRUB2 How To (5): Buil…
Linux From Scratch for …
Ec2 linux server s… 1

GRUB2 How To (… 2
Linux From Scratch for …
Linux Virtual Console(6)…

GRUB2 How To (3) : UE…
Linux From Scratch for …
dnscrypt proxy in Archli…

GRUB2 How To (… 4
Linux From Scratch for …
Git as I understand (9): …

Debugging with QEMU …
Linux From Scratch for …
Linux Virtual Console(5)…

GRUB2 How To (1) : Ma…

Git as I understand (8): …

QEMU 1.5 build configur…
Linux From Scratch for …
Git as I understand (6): …

Linux From Scratch for …
Git as I understand (7): …

Linux From Scratch for …
Git as I understand (5): …

Linux From Scratch for …
Git as I understand (4): …

Linux From Scratch for …
Git as I understand (3): …

Linux From Scratch for …
Git as I understand (2): r…

Linux Virtual Console(4)…
Git as I understand (1): …

Linux Virtual Console(3)…
Mac book pro 11.3 Linux…

Regular Expression: A …
Mac book pro 11.3 Linux…

Automatic Scripts… 2
Mac book pro 11.3 Linux…

```
[luke@rmbp parent]$ git b
  master
* projA
[luke@rmbp parent]$ git l
12fd0d0 a4
1c8922f a3
9f527d9 a2
ca76a50 a1
[luke@rmbp parent]$ git merge -s recursive -X subtree=projA --no-commit --
squash master
Squash commit -- not updating HEAD
Automatic merge went well; stopped before committing as requested
[luke@rmbp parent]$ ls
a1.txt  a2.txt  a3.txt  a4.txt  m1.txt
[luke@rmbp parent]$ git s
## projA...a/master
A  m1.txt
[luke@rmbp parent]$ git commit -m "revert change back to remote projA"
[projA 388c813] revert change back to remote projA
 1 file changed, 1 insertion(+)
 create mode 100644 m1.txt
[luke@rmbp parent]$ git push a projA:master
Counting objects: 3, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 284 bytes | 0 bytes/s, done.
Total 3 (delta 1), reused 0 (delta 0)
To /tmp/git/projA/
   12fd0d0..388c813  projA -> master
```

5. submodule support in Git

When we employ a subtree project in Git, we are dealing with different branches in Git. sub-project lives in separate branch, and we leverage "git rm/read-tree" or "git read-tree/merge -X subtree" to exchange information with imported sub-project.
"git submodule" provides another approach to handle sub project. This time, sub-project is store in separate repository. Sub-project wise, it does not feel any difference when it acts as a sub-project. We operate in sub-project repository just as we do in a normal git repository. Since sub-project repository lives in a sub directory of parent project repository, git needs to provide extra information for parent repository so we do operation in parent repository, git will know how to handle the sub directory with a repository in it.
A typical workflow of "git submodule" is like this:

a. add submodule definition in git

Mac book pro 11.3 Linux…
How to debug system p…
Linux Virtual Console (1…
Mac book pro 11.3 Linux…
How to Bootstrap differe…
How to set up Sof… 2
Mac book pro 11.3 Linux…
DELL R720 iDRAC7 fir…
GRUB2 How To (7): De…
Mac book pro 11.3 Linux…
DELL R710 iDRAC6 set…
GRUB2 How To (6): PX…
Mac book pro 11.3 Linux…
The best way to configur…
GRUB2 How To (5): Buil…
Linux From Scratch for …
Ec2 linux server s… 1
GRUB2 How To (… 2
Linux From Scratch for …
Linux Virtual Console(6)…
GRUB2 How To (3) : UE…
Linux From Scratch for …
dnscrypt proxy in Archli…
GRUB2 How To (… 4
Linux From Scratch for …
Git as I understand (9): …
Debugging with QEMU …
Linux From Scratch for …
Linux Virtual Console(5)…
GRUB2 How To (1) : Ma…
Git as I understand (8): …
QEMU 1.5 build configur…
Linux From Scratch for …
Git as I understand (6): …
Linux From Scratch for …
Git as I understand (7): …
Linux From Scratch for …
Git as I understand (5): …
Linux From Scratch for …
Git as I understand (4): …
Linux From Scratch for …
Git as I understand (3): …
Linux From Scratch for …
Git as I understand (2): r…
Linux Virtual Console(4)…
Git as I understand (1): …
Linux Virtual Console(3)…
Mac book pro 11.3 Linux…
Regular Expression: A …
Mac book pro 11.3 Linux…
Automatic Scripts… 2
Mac book pro 11.3 Linux…

```
## Initial commit on master
A  .gitmodules

A  projA


[luke@rmbp parent]$ cat .gitmodules
[submodule "projA"]
 path = projA
 url = /tmp/git/projA


[luke@rmbp parent]$ cd projA/
[luke@rmbp projA]$ la
.  ..  a1.txt  a2.txt  .git
[luke@rmbp projA]$ cat .git
gitdir: ../.git/modules/projA
[luke@rmbp projA]$ tree ../.git/modules/projA
../.git/modules/projA
├── branches
├── config
├── description
├── HEAD
├── hooks
│   ├── applypatch-msg.sample
.....
├── index
├── info
│   └── exclude
├── logs
│   ├── HEAD
│   └── refs
│       ├── heads
│       │   └── master
│       └── remotes
│           └── origin
│               └── HEAD
├── objects
│   ├── 09
│   │   └── efc7e59a839528ac7bda9fa020dc9101278680
│   ├── 61
...........

[luke@rmbp projA]$ git l

99a71cf a2

61a0159 a1
```

```
 bare = false
 logallrefupdates = true
 worktree = ../../../projA
[remote "origin"]
 url = /tmp/git/projA
 fetch = +refs/heads/*:refs/remotes/origin/*
[branch "master"]
 remote = origin
 merge = refs/heads/master
```

Many interesting things happen after we execute "git submodule add". First, a ".gitmodule" is added into work tree and index, describing the basic information about the added submodule. Second, a sub directory "projA" is added into both work tree and index. Under this sub directory, we can have .git file specifying a git repository path "../.git/modules/projA". We can see git have pull from remote repository all the history of sub project "projA". Thirdly, when we check the config file of this newly create repository, we can see git also setup corresponding remote for this repository. "git submodule add" does lots of necessary setup for our sub-project repository. It save us lots of manual typing.

The most interesting thing is, When we check the index, we found the sub directory "projA" has a new mode "160000" comparing to an ordinary directory tree entry:

```
[luke@rmbp parent]$ git ls-files --stage
100644 d7585a3154fdd162ef79f8a65b1822d0791ad7f6 0 .gitmodules
160000 99a71cf1c6d61ac5f31b75ab5a0f76e383eb1342 0 projA
```

For normal tree entries like the one shown below (testdir), it will have a mode "040000", and generally we can see files under such dir.

```
[luke@rmbp c]$ git ls-tree -tr master
100644 blob ce013625030ba8dba906f756967f9e9ca394464a hello.txt
040000 tree 2b297e643c551e76cfa1f93810c50811382f9117 testdir
100644 blob 9daeafb9864cf43055ae93beb0afd6c7d144bfa4 testdir/test.txt
```

Now let up commit pending change in index to have a more thorough research.

```
[luke@rmbp parent]$ git commit -m "add submodule projA"
[master (root-commit) 9526d3e] add submodule projA
 2 files changed, 4 insertions(+)
 create mode 100644 .gitmodules
 create mode 160000 projA

[luke@rmbp parent]$ tree

.
```

Linux Virtual Console(2)…
Mac book pro 11.3 Linux…
How to debug system p…

Linux Virtual Console (1…
Mac book pro 11.3 Linux…
How to Bootstrap differe…

How to set up Sof… 2
Mac book pro 11.3 Linux…
DELL R720 iDRAC7 fir…

GRUB2 How To (7): De…
Mac book pro 11.3 Linux…
DELL R710 iDRAC6 set…

GRUB2 How To (6): PX…
Mac book pro 11.3 Linux…
The best way to configur…

GRUB2 How To (5): Buil…
Linux From Scratch for …
Ec2 linux server s… 1

GRUB2 How To (… 2
Linux From Scratch for …
Linux Virtual Console(6)…

GRUB2 How To (3) : UE…
Linux From Scratch for …
dnscrypt proxy in Archli…

GRUB2 How To (… 4
Linux From Scratch for …
Git as I understand (9): …

Debugging with QEMU …
Linux From Scratch for …
Linux Virtual Console(5)…

GRUB2 How To (1) : Ma…

Git as I understand (8): …

QEMU 1.5 build configur…
Linux From Scratch for …
Git as I understand (6): …

Linux From Scratch for …
Git as I understand (7): …

Linux From Scratch for …
Git as I understand (5): …

Linux From Scratch for …
Git as I understand (4): …

Linux From Scratch for …
Git as I understand (3): …

Linux From Scratch for …
Git as I understand (2): r…

Linux Virtual Console(4)…
Git as I understand (1): …

Linux Virtual Console(3)…
Mac book pro 11.3 Linux…

Regular Expression: A …
Mac book pro 11.3 Linux…

Automatic Scripts… 2
Mac book pro 11.3 Linux…

```
100644 blob d7585a3154fdd162ef79f8a65b1822d0791ad7f6 .gitmodules
160000 commit 99a71cf1c6d61ac5f31b75ab5a0f76e383eb1342 projA
```

We can see this new entry corresponding to our sub-project is actually a commit object.

```
[luke@rmbp parent]$ cd projA/
[luke@rmbp projA]$ git cat-file -t 99a71cf1c6d61ac5f31b75ab5a0f76e383eb1342
commit
[luke@rmbp projA]$ git cat-file -p 99a71cf1c6d61ac5f31b75ab5a0f76e383eb1342
tree 09efc7e59a839528ac7bda9fa020dc9101278680
parent 61a01592f1a87a7a7eb69a28ea2c00e48aaa6a9c
author Luke Luo <luke.jf.luo@gmail.com> 1403255286 +0800
committer Luke Luo <luke.jf.luo@gmail.com> 1403255286 +0800

a2
[luke@rmbp projA]$ git b
* master
[luke@rmbp projA]$ git rev-parse master
99a71cf1c6d61ac5f31b75ab5a0f76e383eb1342
```

In our sub repository, we can verify this commit is exactly the master HEAD/tip in sub-project. We all know a commit object will point to a tree object. This time, git developer use an uncanny knack to embed an commit object into a tree object. Such special commit within a tree object is called "getlink" in git terminology. Via the help of "getlink", git parent repository can identify a sub repository residing under a sub directory, and act on it accordingly. This is the most important secret behind "git submodule"!

Finally, we can check the status of all submodules via "git submodule status":

```
[luke@rmbp parent]$ git submodule status

 99a71cf1c6d61ac5f31b75ab5a0f76e383eb1342 projA (heads/master)
```

b. init/deinit a submodule

init will add submodule information into parent repository config file, so submodule commands like "update/sync" will act on it. "deinit" will do the reverse:

```
[luke@rmbp parent]$ git submodule init projA
Submodule 'projA' (/tmp/git/projA) registered for path 'projA'
[luke@rmbp parent]$ cat .git/config
[core]
 repositoryformatversion = 0
 filemode = true
 bare = false
 logallrefupdates = true
[submodule "projA"]
```

```
repositoryformatversion = 0
filemode = true
bare = false
logallrefupdates = true
```

c. update

When we add a new repository, the current sub-project branch's tip/HEAD commit is recorded into the sub-directory entry(i.e. projA) in the root tree. After we commit this sub directory, git will remember on which commit we target our sub-module onto. In the submodule, we can pull update from upstream, we can change to different branch and add new commits onto it. All in all, we might change the current branch's tip to other commit other than the one recorded in the submodule dir entry. Whenever we want to checkout the submodule code to our recorded snapshot commit, we can do a "git submodule update". After that, the submodule will be checkout per this commit into a "detach head" state. So "git submodule update" is like a time machine which revert back to our remembered commit.

```
[luke@rmbp parent]$ cd projA
[luke@rmbp projA]$ git l
68a2969 a4
62e4ce7 a3
99a71cf a2
61a0159 a1
[luke@rmbp projA]$ cd ../../projA/
[luke@rmbp projA]$ pwd
/tmp/git/projA
[luke@rmbp projA]$ git simple-commit a5 a6
[master ef4ad2c] a5
 1 file changed, 1 insertion(+)
 create mode 100644 a5.txt
[master ee0b527] a6
 1 file changed, 1 insertion(+)
 create mode 100644 a6.txt
[luke@rmbp projA]$ cd /tmp/git/parent/projA/
[luke@rmbp projA]$ git pull
remote: Counting objects: 6, done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 6 (delta 2), reused 0 (delta 0)
Unpacking objects: 100% (6/6), done.
From /tmp/git/projA
   68a2969..ee0b527  master     -> origin/master
Updating 68a2969..ee0b527
Fast-forward
 a5.txt | 1 +
 a6.txt | 1 +
 2 files changed, 2 insertions(+)
 create mode 100644 a5.txt
```

```
[luke@rmbp projA]$ git l


ee0b527 a6


ef4ad2c a5
68a2969 a4
62e4ce7 a3
99a71cf a2
61a0159 a1
[luke@rmbp parent]$ git diff
diff --git a/projA b/projA
index 68a2969..ee0b527 160000
--- a/projA
+++ b/projA
@@ -1 +1 @@
-Subproject commit 68a29692ad433c9bbfff34a65c6a47f757d231d1
+Subproject commit ee0b527ae643d768e8feca50cbd8431fbdedaab6
[luke@rmbp parent]$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

 modified:   projA (new commits)
```

Here we update upstream projA by adding two extra files "a5.txt/a6.txt". Then within our submodule repository, we do "git pull" to sync it to remote upstream repository. At this time, the current branch is "master" and tip commit is "ee0b527". "git submodule status" will identify this discrepancy between remembered commit and current branch tip in submodule, and a "+" sign indicate such discrepancy. git diff or git status also illustrate such discrepancy in their way. At this time, we have two options:

. revert submodule back to remembered commit via "git submodule update"

```
[luke@rmbp parent]$ git submodule update projA
Submodule path 'projA': checked out '68a29692ad433c9bbfff34a65c6a47f757d231d1'
[luke@rmbp parent]$ git submodule status
 68a29692ad433c9bbfff34a65c6a47f757d231d1 projA (68a2969)
[luke@rmbp parent]$ git diff
[luke@rmbp parent]$ git status
On branch master


nothing to commit, working directory clean
[luke@rmbp parent]$ cd projA/
[luke@rmbp projA]$ git b
* (detached from 68a2969)
  master
[luke@rmbp projA]$ git l
68a2969 a4
```

any more, and "git diff/status" will show a clean state.

. update the remembered commit in sub-project entry and commit it, so we will use this new commit for subproject from now on:

```
[luke@rmbp parent]$ git submodule status
+ee0b527ae643d768e8feca50cbd8431fbdedaab6 projA (heads/master)
[luke@rmbp parent]$ git ls-files -s
100644 88f27c789175da080dcf027afea293b2388e0b1e 0 .gitmodules
160000 68a29692ad433c9bbfff34a65c6a47f757d231d1 0 projA
[luke@rmbp parent]$ git add projA/
[luke@rmbp parent]$ git ls-files -s
100644 88f27c789175da080dcf027afea293b2388e0b1e 0 .gitmodules
160000 ee0b527ae643d768e8feca50cbd8431fbdedaab6 0 projA
[luke@rmbp parent]$ git commit -m "move to new commit for submodule"
[master 5858b22] move to new commit for submodule
 1 file changed, 1 insertion(+), 1 deletion(-)
[luke@rmbp parent]$ git submodule status
 ee0b527ae643d768e8feca50cbd8431fbdedaab6 projA (heads/master)
```

Posted 20th June 2014 by Luke Luo

0 Add a comment