Git as I understand (9): Patch

Git is designed as a snapshot based object store, patch(or diff) is NOT a native object. When we need a diff/patch of different nodes, we ask git to compute it on the fly. When it comes to remote collaboration, if both sides employ git as version control, we can use "git merge" to exchange information. But interestingly, many projects still employ patch file as a tool for code delivery, including Linux Kernel. Per my understanding, using patch as delivery mechanism has below advantages:

a. For those who did not use git as version control, diff/patch is still a valid universal tools;
b. For code review, developers often tend to review a patch file. Such a patch file provides more necessary context for an efficient code review. A patch file is just a text file, you can view it even if you don't use git.
c. git provides extra support for patch when comparing to traditional "diff/patch" program. One of the critical extra support is "git diff/format-patch" record the object id of involved tree/blog objects. When you apply patch into git again, such information would make a 3-way merge possible. A traditional patch will just fail if patch can not apply cleanly;
d. "git format-patch" will record complete information for a commit. So when you apply the patch via "git am",all commit history will be preserved in addition to code change.

There are two pairs of git command to generate and apply patch. They are "git diff/apply" and "git format-patch/am".

1. git diff/apply

This works just like traditional "diff/patch". The patch generated only reflects difference between two commits. All interim commits change will be squash into one patch file. The git diff always generate diff from the root tree of a commit. You don't need to align the directory level when you do "git apply", as you generally did in a traditional "patch -pN".

When you apply this patch, index or work tree (or both) are updated. You need to commit these change to form a totally new commit. The original commit history is not brought into your new commit. Since the object id is also carried with the diff, when a patch does not apply cleanly, it is possible to resort back to a "3 way merge" via "git apply --3way".

2. git format-patch/am

When you use "git format-patch" , a series of patch files will be generated, one per commit you selected. In addition to the code change, the commit information is also included into the generated patch file, like author, commit

3. patch SHA1 id: git patch-id

Git allows you to compute a SHA1 id for a diff/patch. If two commits have identical patch SHA1 id, we can basically ensure that they provide identical contribution to repository. Such case happens when:

a. You cherry-pick a commit from another branch. So the patch SHA1 of these different commits are the same;
b. you apply a patch in your branch. When compare the patch id of generated commit in your branch versus the original commit which generate the patch, these two patch id are the same

Here is an example:

```
[luke@rmbp patchid]$ git b
  left
  master
* right
[luke@rmbp patchid]$ git lg
* edc7654 (HEAD, right) r3
* 0e2ca07 r2
* cdf1d1f r1
| * 89e4290 (left) l2
| * e0c5392 l1
|/
* 9be22c6 (master) m2
* 875d734 m1
[luke@rmbp patchid]$ git diff right^ right
diff --git a/m1.txt b/m1.txt
index 63a911f..61c5d64 100644
--- a/m1.txt
+++ b/m1.txt
@@ -1 +1,2 @@
 m1
+change to introduce patch-id
diff --git a/r3.txt b/r3.txt
new file mode 100644
index 0000000..b6693b6
--- /dev/null
+++ b/r3.txt
@@ -0,0 +1 @@
+r3
[luke@rmbp patchid]$ git diff right^ right | git patch-id
4479a1c0b3cf4872d3ab0190e99553e541c8ec09
0000000000000000000000000000000000000000
```

I have there branches (master/left/right). In the right branch, I generate a patch id for commit "r3"

Dynamic Views theme. Powered by Blogger.

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

Git as I understand (9): …

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

GRUB2 How To (1) : Ma…
Linux From Scratch for …
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 patchid]$ git s
## left
M  m1.txt
A  r3.txt
[luke@rmbp patchid]$ git commit -m "l3"
[left 5928e12] l3
 2 files changed, 2 insertions(+)
 create mode 100644 r3.txt
[luke@rmbp patchid]$ git diff left^ left
diff --git a/m1.txt b/m1.txt
index 63a911f..61c5d64 100644
--- a/m1.txt
+++ b/m1.txt
@@ -1 +1,2 @@
 m1
+change to introduce patch-id
diff --git a/r3.txt b/r3.txt
new file mode 100644
index 0000000..b6693b6
--- /dev/null
+++ b/r3.txt
@@ -0,0 +1 @@
+r3
[luke@rmbp patchid]$ git diff left^ left | git patch-id
4479a1c0b3cf4872d3ab0190e99553e541c8ec09
0000000000000000000000000000000000000000
```

We cherry-pick commit "r3" into branch "left" and generate a new commit "l3". We have verified above the patch SHA1 id of "r3" and "l3" are identical (**`4479a1c0b3cf4872d3ab0190e99553e541c8ec09`**)**!**

You can also appy commit "r3" on top of branch "left". The result is still the same.
"git rebase" apply the "patch-id trick" to help to identify the commits which already have been applied onto upstream. Quotation from "git help rebase":

"The commits that were previously saved into the temporary area are then reapplied to the current branch, one by one, in order. ***Note that any commits in HEAD which introduce the same textual changes as a commit in HEAD..<upstream> are omitted*** *(i.e., a patch already accepted upstream with a different commit message or timestamp will be* ***skipped****)."*

"git cherry" is another command to help to do "git rebase". It will identify the commits in your feature branch which have been applied in upstream branch via "git cherry-pick" or patch application. For example, when I want to rebase my "right" branch into "left" branch, "git cherry" will tell me "r3" has

Dynamic Views theme. Powered by Blogger.

```
edc7654 r3
0e2ca07 r2
cdf1d1f r1
9be22c6 m2
875d734 m1
[luke@rmbp patchid]$ git cherry left right
+ cdf1d1f049b5488dcd9a9c790eb57d8a9a9ba7a9
+ 0e2ca071fdebe29d207cf9908c10bff92592eb9b
- edc765446270655cb97a79422e38c7363eeac4f0
```

Here a "-" sign indicates this commit has already applied into upstream. In "git rebase", this commit ("r3") will be skipped.

Posted 23rd June 2014 by Luke Luo

0 Add a comment

Enter your comment...

Comment as: bharathvenna.v Sign out

Publish Preview Notify me