22 MARCH 2012

Git Cherry

Sometimes overuse of git cherry-pick command leads to the situation, where actual difference between two branches is no longer the one you see in git `log branch1..branch2`.

For example, if repo has two branches: `master` and `experimental`. `Master`'s history looks like this:

```
~/example> git checkout master
~/example> git log --oneline
4c72a0a First commit
24361e1 Initial commit
```

And experimental's:

```
~/example> git checkout experimental
~/example> git log --oneline
31e76c6 Second experimental commit
160a97c First experimental commit
4c72a0a First commit
24361e1 Initial commit
```

Now let's cherry-pick one commit to the master branch:

```
~/example> git checkout master
~/example> git cherry-pick 160a97c
~/example> git log --oneline
f16b462 First experimental commit
4c72a0a First commit
24361e1 Initial commit
```

Note that new commit has a different SHA-1 now, because `cherry-pick` "applies changes introduced by some existing commits", not moves commit object itself. So now if we try to see what's new in `experimental` branch compared to `master` brach using `git log`, we get this:

```
~/example> git log master..experimental --oneline --graph --pretty=format:'%H - %s'
* 31e76c6893f01283bae005c9af1a25a7c63b1423 - Second experimental commit
* 160a97cd0b1a32381e34ea5f72f5f39cacb77e14 - First experimental commit
```

There's no difference between "First experimental commit" and "Second experimental commit" though changes presented by one of them is already in the `master`. But `git cherry` command can show that difference.

```
~/example> git cherry master experimental -v
- 160a97cd0b1a32381e34ea5f72f5f39cacb77e14 First experimental commit
+ 31e76c6893f01283bae005c9af1a25a7c63b1423 Second experimental commit
```

`+` means that "upstream" (in this case `master` branch) lacks both commit object and it's changes. `-` tells us that changes of this commit are represented by a different commit object in "upstream".

Now a little more about command syntax:

The first argument (optional, defaults to the branch's remote) is *upstream*. There's a lot of confusion about this term, so I would recommend this question on StackOverflow. In this case upstream is simply a branch in which you want to know what commits from another branch (head) are present. The second argument "head" is also optional and defaults to current HEAD.