沙耶

git merge简介

git merge的基本用法为把一个分支或或某个commit的修改合并到现在的分支上。
我们可以运行git merge -h和git merge --help查看其命令，后者会直接转到一个网页（git的帮助文档），更详细。

```
usage: git merge [options] [<commit>...]
   or: git merge [options] <msg> HEAD <commit>
   or: git merge --abort

    -n                    do not show a diffstat at the end of the merge
    --stat                show a diffstat at the end of the merge
    --summary             (synonym to --stat)
    --log[=<n>]           add (at most <n>) entries from shortlog to merge
commi                                                              t message
    --squash              create a single commit instead of doing a merge
    --commit              perform a commit if the merge succeeds (default)
    -e, --edit            edit message before committing
    --ff                  allow fast-forward (default)
    --ff-only             abort if fast-forward is not possible
    --rerere-autoupdate   update the index with reused conflict resolution if
po                                                                  ssible
    -s, --strategy <strategy>
                          merge strategy to use
    -X, --strategy-option <option=value>
                          option for selected merge strategy
    -m, --message <message>
                          merge commit message (for a non-fast-forward merge)
    -v, --verbose         be more verbose
    -q, --quiet           be more quiet
    --abort               abort the current in-progress merge
    --progress            force progress reporting
    -S, --gpg-sign[=<key id>]
                          GPG sign commit
    --overwrite-ignore    update ignored files (default)
```

git merge [options] <msg> HEAD <commit> 这里的 HEAD 其实就是分支名，用于说明把 HEAD 分支合并到当前分支。

--squash选项的含义是：本地文件内容与不使用该选项的合并结果相同，但是不保留待合并分支上的历史信息，也不提交、不移动HEAD，因此需要一条额外的commit命令。其效果相当于将another分支上的多个commit合并成一个，放在当前分支上，原来的commit历史则没有拿过来。
判断是否使用--squash选项最根本的标准是，待合并分支上的历史是否有意义。
如果在开发分支上提交非常随意，甚至写成微博体，那么一定要使用--squash选项。版本历史记录的应该是代码的发展，而不是开发者在编码时的活动。
只有在开发分支上每个commit都有其独自存在的意义，并且能够编译通过的情况下（能够通过测试就更完美了），才应该选择缺省的合并方式来保留commit历史。
--no-ff
Create a merge commit even when the merge resolves as a fast-forward. This is the default behaviour when merging an annotated (and possibly signed) tag.
我们在将Develop分支发布到Master分支时，可能采用如下的命令：

```
    # 切换到Master分支
    git checkout master
    # 对Develop分支进行合并
```

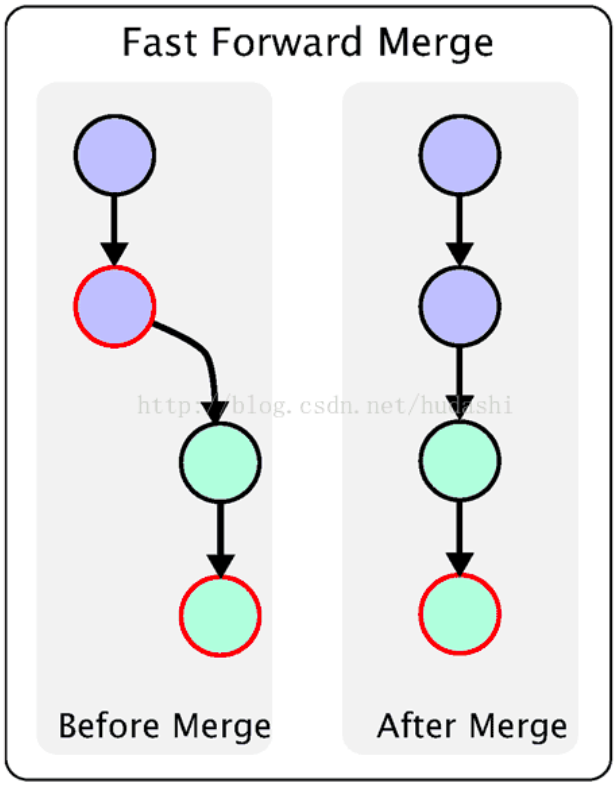
git merge --no-ff develop

这里稍微解释一下，上一条命令的--no-ff参数是什么意思。

这个命令要通过git merge --help查看其命令，后者会直接转到一个网页，才能看到其详细说明

默认情况下，Git执行"快进式合并"（fast-farward merge），会直接将Master分支指向Develop分支。

示图1-1



示图1-2

命令：

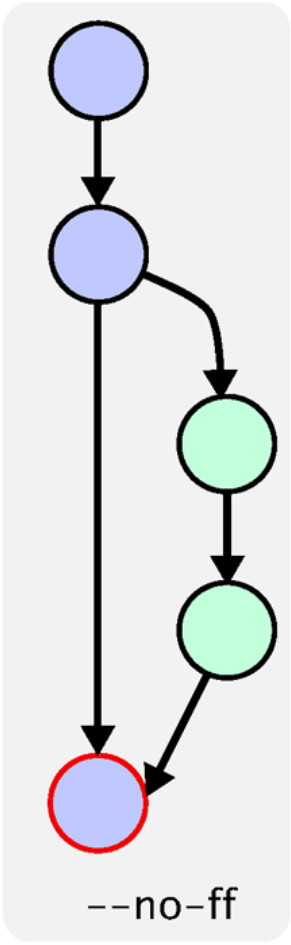
git checkout master

git merge robin_local

使用--no-ff参数后，会执行正常合并，在Master分支上生成一个新节点。为了保证版本演进的清晰，我们希望采用这种做法。关于合并的更多解释，请参考Benjamin Sandofsky的《Understanding the Git Workflow》。

示图2-1

随笔档案



示图2-2
命令：
git checkout master
git merge robin_local

图3

以下是一篇来自于哈佛大学关于git merge的文章
英文地址： http://www.eecs.harvard.edu/~cduan/technical/git/git-3.shtml

Merging

我的园子

阅读排行榜

推荐排行榜

After you have finished implementing a new feature on a branch, you want to bring that new feature into the main branch, so that everyone can use it. You can do so with the `git merge` or `git pull` command.

The syntax for the commands is as follows:

```
git merge [head]
git pull . [head]
```

They are identical in result. (Though the `merge` form seems simpler for now, the reason for the `pull` form will become apparent when discussing multiple developers.)

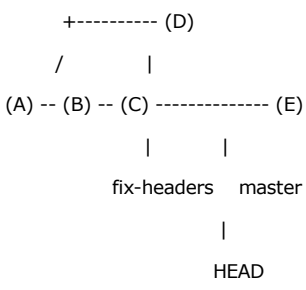
These commands perform the following operations. Let the current head be called current, and the head to be merged calledmerge.

1. Identify the common ancestor of current and merge. Call it ancestor-commit.
2. Deal with the easy cases. If the ancestor-commit equals merge, then do nothing. If ancestor-commit equals current, then do a fast forward merge.
3. Otherwise, determine the changes between the ancestor-commit and merge.
4. Attempt to merge those changes into the files in current.
5. If there were no conflicts, create a new commit, with two parents, current and merge. Set current (and HEAD) to point to this new commit, and update the working files for the project accordingly.
6. If there was a conflict, insert appropriate conflict markers and inform the user. No commit is created.

Important note: Git can get very confused if there are uncommitted changes in the files when you ask it to perform a merge. So make sure to commit whatever changes you have made so far before you merge.

So, to complete the above example, say you check out the master head again and finish writing up the new data for your paper. Now you want to bring in those changes you made to the headers.

The repository looks like this:

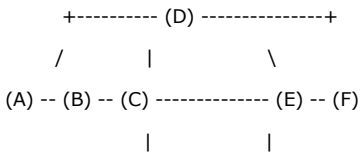


where (E) is the commit reflecting the completed version with the new data.

You would run:

```
git merge fix-headers
```

If there are no conflicts, the resulting respository looks like this:



```
          fix-headers          master
                                 |
                                HEAD
```

The merge commit is (F), having parents (D) and (E). Because (B) is the common ancestor between (D) and (E), the files in (F) should contain the changes between (B) and (D), namely the heading fixes, incorporated into the files from (E).

Note on terminology: When I say "merge head A into head B," I mean that head B is the current head, and you are drawing changes from head A into it. Head B gets updated; nothing is done to head A. (If you replace the word "merge" with the word "pull," it may make more sense.)

Resolving Conflicts

A conflict arises if the commit to be merged in has a change in one place, and the current commit has a change in the same place. Git has no way of telling which change should take precedence.

To resolve the commit, edit the files to fix the conflicting changes. Then run `git add` to add the resolved files, and run`git commit` to commit the repaired merge. Git remembers that you were in the middle of a merge, so it sets the parents of the commit correctly.

如果没有冲突的话，merge完成。有冲突的话，git会提示那个文件中有冲突，比如有如下冲突：

<<<<<<< HEAD:test.c

printf ("test1");

=======

printf ("test2");

>>>>>>> issueFix:test.c

可以看到 ======= 隔开的上半部分，是 HEAD（即 master 分支，在运行 merge 命令时检出的分支）中的内容，下半部分是在 issueFix 分支中的内容。解决冲突的办法无非是二者选其一或者由你亲自整合到一起。比如你可以通过把这段内容替换为下面这样来解决：

printf ("test2");

这个解决方案各采纳了两个分支中的一部分内容，而且删除了 <<<<<<<，=======，
和>>>>>>> 这些行。

在解决了所有文件里的所有冲突后，运行git add 将把它们标记为已解决（resolved）。

然后使用git commit命令进行提交，merge就算完成了

Fast Forward Merges

A fast forward merge is a simple optimization for merging. Say your repository looks like this:

```
          +-- (D) ------ (E)
            /            |
(A) -- (B) -- (C)        |
            |            |
          current     to-merge
            |
           HEAD
```

and you run `git merge to-merge`. In this case, all Git needs to do is set current to point to (E). Since (C) is the common ancestor, there are no changes to actually "merge."

Hence, the resulting merged repository looks like:

```
            +-- (D) -- (E)
           /          |
(A) -- (B) -- (C)     |
                      |
                 to-merge, current
                      |
                    HEAD
```

That is, to-merge and current both point to commit (E), and HEAD still points to current.

Note an important difference: no new commit object is created for the merge. Git only shifts the head pointers around.

Common Merge Use Patterns

There are two common reasons to merge two branches. The first, as explained above, is to draw the changes from a new feature branch into the main branch.

The second use pattern is to draw the main branch into a feature branch you are developing. This keeps the feature branch up to date with the latest bug fixes and new features added to the main branch. Doing this regularly reduces the risk of creating a conflict when you merge your feature into the main branch.

One disadvantage of doing the above is that your feature branch will end up with a lot of merge commits. An alternative that solves this problem is rebasing , although that comes with problems of its own.

分类：Git

· 盖茨：购买比特币属于"博傻"投资 我要做空它
· 《王者荣耀》负优化索要巨额保护费？腾讯回应
· Windows 10 4月更新Bug汇总：步步为坑
· 《腾讯没有梦想》作者：文章阅读量破百万 腾讯仍有许多问题需解决
· 阿里影业公布2017/2018财年业绩：营收超33亿
» 更多新闻...

最新知识库文章:
· 如何成为优秀的程序员?
· 菜鸟工程师的超神之路 -- 从校园到职场
· 如何识别人的技术能力和水平?
· 写给自学者的入门指南
· 和程序员谈恋爱
» 更多知识库文章...

历史上的今天:
2014-03-08 由CAST()函数在.NET1.1和.NET4.0下处理机制不同所引发的BUG