GIT 操作手册

假设大家已经:

- 配置好了账号和key
- 编辑好了.gitconfig
- repo clone到了本地

Review 流程

1. 首先把修改提交到本地repo。

   ```
   git ci
   ```

   如果是针对刚才ci的修补，请使用:

   ```
   git ca
   ```

2. 同步remote repo，如果没有更新请直接跳到5:

   ```
   git fetch
   ```

3. 将本地的commit rebase到远程master最后。origin/master请修改为正确的远程分枝. 如果没有冲突请直接跳到5

   ```
   git rebase origin/master
   ```

4. 解决冲突之后执行以完成rebase

   ```
   git rebase --continue
   ```

5. 提交review.

   ```
   git review
   ```

   注意如果是合并到其它branch的review, 请参考.gitconfig中review的定义使用push命令, 把master修改为正确的branch再提交:

   ```
   git push origin HEAD:refs/for/refs/heads/master
   ```

6. 收到修改的comment并修改，回到1(使用git ca)重新执行流程

PS1: 步骤1提交之前还需要git add等操作

PS2: 步骤2用git pull也可以。区别在于pull会把local master和origin/master合并，而fetch只是单纯的同步remote repo。

PS3: 步骤3主要目的是让commit是base在origin/master最新的change之上。也可以解决步骤2使用git pull产生的Merger commit没有Change-Id导致review不能提交的问题。

Q&A

review提交失败

1. git pull产生的Merge Commit没有Change-Id, 请使用rebase消除Merge Commit后重新提交review
2. 没有scp Change-Id生成脚本，需要review的commit没有Change-Id, 请参考提示：
   - 将Change-Id手动添加到comment里;
   - 执行提示里的命令scp Change-Id生成脚本，下一次的ci会自动生成comment.
3. 当前review已经关闭（Abandoned or Submitted）请按如下执行:

```
git fetch
git reset origin/master # 请把master换成正确的分支
git st # 这一步应该能查看到一些没提交的修改
git add ... # ...请换成添加需要提交的文件
git ci
```

4. 配置的用户名不正确或者在没有配置用户名的时候提交了ci，请检查修正.gitconfig配置后，参考3执行

review出了多个Change

多次使用ci形成了多个commit, 每个commit会对应一个Change. 请使用“合并commit"将多个commit合并为一个后重新review, 并abandon多余的review.

删除无用的commit

使用git rebase -i, 具体请参考"合并commit"的最后一个栗子

合并commit

1. 如果是最后几个commit需要合并，请按如下执行：

```
git reset HEAD~1 # ～1适用于合并最后两个commit. 合并最后N个commit请把1改成N-1
git add ... # ...请换成添加需要提交的文件
git ca # amend添加的修改
```

2. 使用rebase合并多个commit. 注意，请不要合并从remote repo同步到的commit.

```
git rebase HEAD~3 # 对最后的3个commit操作合并
```

会进入编辑器，内容是最后3次commit, 请注意顺序与git log的顺序相反，旧的在上新的在下：

```
pick 14ff8b8 add lang        # a
pick 1efe707 add linux       # b
pick 711868b add db          # c
```

把需要合并的几个commit中较新（靠下）的commit前面的"pick"修改为"squash",

- 如果需要合并 a, b:

```
pick 14ff8b8 add lang        # a
squash 1efe707 add linux     # b
pick 711868b add db          # c
```

- 如果需要合并 b, c:

```
pick 14ff8b8 add lang        # a
pick 1efe707 add linux       # b
squash 711868b add db        # c
```

- 如果需要合并 a, b, c:

```
pick 14ff8b8 add lang        # a
squash 1efe707 add linux     # b
squash 711868b add db        # c
```

- 也可以删除某些commit, 比如删除b, 合并a和c:

```
pick 14ff8b8 add lang        # a
drop 1efe707 add linux       # b
squash 711868b add db        # c
```

查看修改历史

- git log -1

查看最后1条记录，1可以改成任意数字

- git hist

  # 查看merge的合并图示

- git log --name-status

  # 在log中查看每个commit变动的文件

- git difftool

  # 使用工具（通常是vimdiff）查看diff

- git blame FILE

  # 查看文件的修改记录

临时转移工作目录的修改

- git stash

  # 把工作目录中的修改暂存转移，工作目录会回到没有修改的状态，方便进行其它git操作

- git stash pop

  # 把暂存的修改恢复到工作目录

- git stash drop

  # 把暂存的修改删除（不可恢复）

- git stash的其它操作请参考 man git-stash

git rebase的一些说明

- rebase相当于把一组commit的diff拿出来，重新ci到指定的change之后，形成一组新的commit.
- rebase之后的commit-id与rebase之前是不一样的。
- 对于git rebase -i
  - squash就是把多个diff合并后commit.
  - drop直接把这个diff放弃不ci. 删除掉这一行也有同样的效果
  - 调整pick的顺序就是调整ci的顺序

常用alias

```
[alias]
 co = checkout
 ci = commit
 ca = commit --amend
 st = status
 fr = !git fetch && git rebase
 sp = !git stash && git pull && git stash pop
 br = branch
 dc = diff --cached
 sst = status -s
 hist = log --pretty=format:\"%h %ad | %s%d [%an]\" --graph --date=short
 review = push origin HEAD:refs/for/refs/heads/master
```