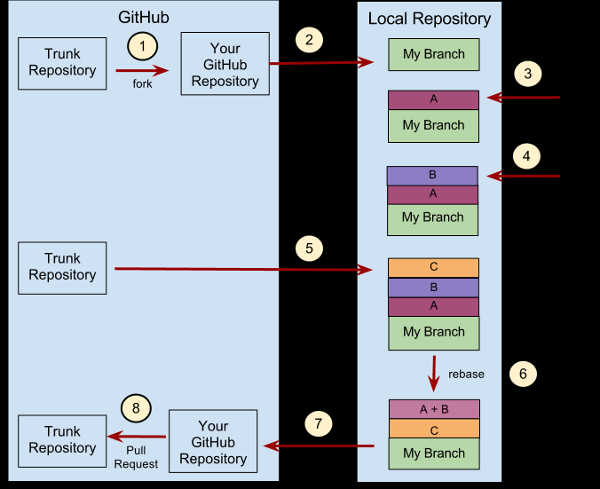
Git 使用规范流程

团队开发中，遵循一个合理、清晰的Git使用流程，是非常重要的。

否则，每个人都提交一堆杂乱无章的commit，项目很快就会变得难以协调和维护。

下面是[ThoughtBot](https://github.com/thoughtbot/guides/tree/master/protocol/git) 的Git使用规范流程。我从中学到了很多，推荐你也这样使用Git。



第一步：新建分支

首先，每次开发新功能，都应该新建一个单独的分支（这方面可以参考[《Git分支管理策略》](http://www.ruanyifeng.com/blog/2012/07/git.html)）。

# 获取主干最新代码

$ git checkout master

$ git pull

# 新建一个开发分支myfeature

$ git checkout -b myfeature

第二步：提交分支commit

分支修改后，就可以提交commit了。

$ git add --all

$ git status

$ git commit --verbose

git add 命令的all参数，表示保存所有变化（包括新建、修改和删除）。从Git 2.0开始，all是 git add 的默认参数，所以也可以用 git add . 代替。

git status 命令，用来查看发生变动的文件。

git commit 命令的verbose参数，会列出 [diff](http://www.ruanyifeng.com/blog/2012/08/how_to_read_diff.html) 的结果。

第三步：撰写提交信息

提交commit时，必须给出完整扼要的提交信息，下面是一个范本。

Present-tense summary under 50 characters

\* More information about commit (under 72 characters).\* More information about commit (under 72 characters).

http://project.management-system.com/ticket/123

第一行是不超过50个字的提要，然后空一行，罗列出改动原因、主要变动、以及需要注意的问题。最后，提供对应的网址（比如Bug ticket）。

第四步：与主干同步

分支的开发过程中，要经常与主干保持同步。

$ git fetch origin

$ git rebase origin/master

第五步：合并commit

分支开发完成后，很可能有一堆commit，但是合并到主干的时候，往往希望只有一个（或最多两三个）commit，这样不仅清晰，也容易管理。

那么，怎样才能将多个commit合并呢？这就要用到 git rebase 命令。

$ git rebase -i origin/master

git rebase命令的i参数表示互动（interactive），这时git会打开一个互动界面，进行下一步操作。

下面采用[Tute Costa](https://robots.thoughtbot.com/git-interactive-rebase-squash-amend-rewriting-history)的例子，来解释怎么合并commit。

pick 07c5abd Introduce OpenPGP and teach basic usage

pick de9b1eb Fix PostChecker::Post#urls

pick 3e7ee36 Hey kids, stop all the highlighting

pick fa20af3 git interactive rebase, squash, amend

# Rebase 8db7e8b..fa20af3 onto 8db7e8b

#

# Commands:

# p, pick = use commit

# r, reword = use commit, but edit the commit message

# e, edit = use commit, but stop for amending

# s, squash = use commit, but meld into previous commit

# f, fixup = like "squash", but discard this commit's log message

# x, exec = run command (the rest of the line) using shell

#

# These lines can be re-ordered; they are executed from top to bottom.

#

# If you remove a line here THAT COMMIT WILL BE LOST.

#

# However, if you remove everything, the rebase will be aborted.

#

# Note that empty commits are commented out

上面的互动界面，先列出当前分支最新的4个commit（越下面越新）。每个commit前面有一个操作命令，默认是pick，表示该行commit被选中，要进行rebase操作。

4个commit的下面是一大堆注释，列出可以使用的命令。

* pick：正常选中
* reword：选中，并且修改提交信息；
* edit：选中，rebase时会暂停，允许你修改这个commit（参考[这里](https://schacon.github.io/gitbook/4_interactive_rebasing.html)）
* squash：选中，会将当前commit与上一个commit合并
* fixup：与squash相同，但不会保存当前commit的提交信息
* exec：执行其他shell命令

上面这6个命令当中，squash和fixup可以用来合并commit。先把需要合并的commit前面的动词，改成squash（或者s）。

pick 07c5abd Introduce OpenPGP and teach basic usage

s de9b1eb Fix PostChecker::Post#urls

s 3e7ee36 Hey kids, stop all the highlighting

pick fa20af3 git interactive rebase, squash, amend

这样一改，执行后，当前分支只会剩下两个commit。第二行和第三行的commit，都会合并到第一行的commit。提交信息会同时包含，这三个commit的提交信息。

# This is a combination of 3 commits.

# The first commit's message is:

Introduce OpenPGP and teach basic usage

# This is the 2nd commit message:

Fix PostChecker::Post#urls

# This is the 3rd commit message:

Hey kids, stop all the highlighting

如果将第三行的squash命令改成fixup命令。

pick 07c5abd Introduce OpenPGP and teach basic usage

s de9b1eb Fix PostChecker::Post#urls

f 3e7ee36 Hey kids, stop all the highlighting

pick fa20af3 git interactive rebase, squash, amend

运行结果相同，还是会生成两个commit，第二行和第三行的commit，都合并到第一行的commit。但是，新的提交信息里面，第三行commit的提交信息，会被注释掉。

# This is a combination of 3 commits.

# The first commit's message is:

Introduce OpenPGP and teach basic usage

# This is the 2nd commit message:

Fix PostChecker::Post#urls

# This is the 3rd commit message:

# Hey kids, stop all the highlighting

[Pony Foo](http://ponyfoo.com/articles/git-github-hacks)提出另外一种合并commit的简便方法，就是先撤销过去5个commit，然后再建一个新的。

$ git reset HEAD~5

$ git add .

$ git commit -am "Here's the bug fix that closes #28"

$ git push --force

squash和fixup命令，还可以当作命令行参数使用，自动合并commit。

$ git commit --fixup

$ git rebase -i --autosquash

这个用法请参考[这篇文章](http://fle.github.io/git-tip-keep-your-branch-clean-with-fixup-and-autosquash.html)，这里就不解释了。

第六步：推送到远程仓库

合并commit后，就可以推送当前分支到远程仓库了。

$ git push --force origin myfeature

git push命令要加上force参数，因为rebase以后，分支历史改变了，跟远程分支不一定兼容，有可能要强行推送（参见[这里](http://willi.am/blog/2014/08/12/the-dark-side-of-the-force-push/)）。

Git 常用命令

git init here -- 创建本地仓库(repository)，将会在文件夹下创建一个 .git 文件夹，.git 文件夹里存储了所有的版本信息、标记等内容

git remote add origin git@github.com:winter1991/helloworld.git

-- 把本地仓库和远程仓库关联起来。如果不执行这个命令的话，每次 push 的时候都需要指定远程服务器的地址

git add -- 从本地仓库增删，结果将会保存到本机的缓存里面

git rm

git commit -m "注释" -- 提交，把本机缓存中的内容提交到本机的 HEAD 里面

git push origin master -- 把本地的 commit(提交) push 到远程服务器上， origin 也就是之前 git remote add origin 那个命令里面的 origin，origin 替代了服务器仓库地址：git push git@github.com:winter1991/helloworld.git master

git pull origin master -- 从远程服务器 pull 新的改动

git status -- 查看状态

git add -A -- 提交全部修改

1. git 配置：

git config --global user.name "xxx" -- 配置用户名，上传本地 repository 到服务器上的时候，在 Github 上会显示这里配置的上传者信息

git config --global user.email "xxx" -- 配置邮箱

git config --list 查看配置列表

配置 sshkey ： 上传代码时使用这个 sshkey 来确认是否有上传权限

1. 创建本地 ssh ： ssh-keygen -t rsa -C "Github 的注册邮箱"2. 在 Github 中添加这个 sshkey ：

复制 C:\Documents and Settings\Administrator\.ssh\id\_rsa.pub 文件中的内容；

登录 Github --> Account Setting --> SSH-KEY --> Add SSH-KEY --> 粘贴id\_rsa.pub中的内容；

3. 验证： ssh -T git@github.com

出现 Hi xxx! You've successfully authenticated, but GitHub does not provide shell access. 说明配置成功，可以连接上 Github2. 建立仓库 repository ：

git init here -- 创建本地仓库

git remote add origin git@github.com:用户名/仓库名.git

-- 把本地仓库和远程仓库关联起来， 如果不执行这个命令的话，每次 push 的时候都需要指定远程服务器的地址

3. 从远程仓库中下载新的改动：

git pull origin master

4. 提交本地修改到远程仓库中：

git add

git add -A -- 将改动添加到本地仓库中

git rm xxx -- 从本地仓库中删除指定文件

git rm -r xxx -- 从本地仓库中删除指定文件夹

git commit -m "注释" -- 把本机缓存中的内容提交到本机的 HEAD 里面

git push origin master -- 把本地的 commit push 到远程仓库中

5. 使用 .gitignore 文件忽略指定的内容：

1. 在本地仓库根目录创建 .gitignore 文件。Win7 下不能直接创建，可以创建 ".gitignore." 文件，后面的标点自动被忽略；

2. 过滤文件和文件夹： [Tt]emp/ 过滤 Temp\temp 文件夹； \*.suo 过滤 .suo 文件；

3. 不过滤文件和文件夹： !\*.c