



神奇辉
404 前端开发

先来一个段子，来自于王垠，新鲜热辣：

<http://www.yinwang.org/blog-cn/2015/03/11/git-etiquette/>

关于Git的礼节

王垠
2015-03-03



ps: 先抛开对王垠的偏见。
毕竟人家是大神。

（这里的内容本来是《怎样尊重一个程序员》的一小节，但由于Git的使用引起了很普遍的不尊重程序员的现象，现在特别将这一节提出来单独成文。）

Git是现在最流行的代码版本控制工具。用外行话说，Git就是一个代码的“仓库”或者“保管”，这样很多人修改了代码之后，可以知道是谁改了哪一块。其实不管什么工具，不管是编辑器，程序语言，还是版本控制工具，比起程序员的核心思想来，都是次要的东西，都是起辅助作用的。可是Git这工具似乎特别惹人恼火。

Git并不像很多人吹嘘的那么好用，其中有明显的蹩脚设计。跟Unix的传统一脉相承，Git没有一个良好的包装，设计者把自己的内部实现细节无情地泄露给了用户，让用户需要琢磨设计者内部到底怎么实现的，否则很多时候不知道该怎么办。用户被迫需要记住挺多稀奇古怪的命令，而且命令行的设计也不怎么合理，有时候你需要加-f之类的参数，各个参数的位置可能不一致，而且加了还不一定能起到你期望的效果。各种奇怪的现象，比如“head detached”，都强迫用户去了解它内部是怎么设计的。随着Git版本的更新，新的功能和命令不断地增加，后来你终于看到命令行里出现了foreach，才发现它的命令行就快变成一个（劣质的）程序语言。如果你了解ydiff的设计思想，就会发现Git之类基于文本的版本控制工具，其实属于古代的东西。然而很多人把Git奉为神圣，就因为它是Linus Torvalds设计的。

Git最让人恼火的地方并不是它用起来麻烦，而是它的“资深用户”们居高临下的态度给你造成的心理阴影。好些人因为自己“精通Git”就以为高人一等，摆出一副专家的态度。随着用户的增加，Git最初的设计越来越被发现不够用，所以一些约定俗成的规则似乎越来越多，可以写成一本书！跟Unix的传统一脉相承，Git给你很多可以把自己套牢的“机制”，到时候出了问题就怪你自己不知道。所以你就经常听有人煞有介事的说：“并不是Git允许你这么做，你就可以这么做的！Unix的哲学是不阻止傻人做傻事.....”如果你提交代码时不知道Git用户一些约定俗成的规则，就会有人嚷嚷：“rebase了再提交！”“不要push到master！”“不要merge！”“squash commits！”如果你不会用git submodule之类的东西，有人可能还会鄙视你，说：“你应该知道这些！”

打个比方，这样的嚷嚷给人的感觉是，你得了奥运会金牌之后，把练习用的器材还回到器材保管科，结果管理员对你大吼：“这个放这边！那个放那边！懂不懂规矩啊你？”看出来问题了吗？程序员提交了有高价值的代码（奥运金牌），结果被一些自认为Git用的很熟的人（器材保管员）厉声呵斥。

一个尊重程序员的公司文化，就应该把程序员作为运动健将，把程序员的代码放在尊贵的地位。其它的工具，都应该像器材保管科一样。我们尊重这些器材保管员，然而如果运动员们不懂你制定的器材摆放规矩，也应该表示出尊重和理解，说话应该和气有礼貌，不应该骑到他们头上。所以，对于Git的一些命令和用法，我建议大家向新手介绍时，这样开场：“你本来不该知道这些的，可是现在我们没有更好的工具，所以得这样弄一下.....”

段子讲完了，也是我今天 showcase 的原因。

- 1、为了避免段子里面的场景，得需要让每个成员了解团队的协助流程。
- 2、更好的推广 ZoomEye 良好的 git 协助，让大家借鉴。
- 3、普及 git 。

记住：

Git 已经成为程序员使用最多的源代码管理工具！
高性能、简单的设计、非线性高并发分支的支持和完全的分布式。
---MacTalk

尊重每个团队的习惯，新成员加入，带领新成员熟悉团队习惯。



Git 简介

Git 是什么鬼？

就一个版本管理工具。

市面上一堆的版本管理工具，大家熟悉的，SVN、GIT、HG，
更古老的，CVS、还有其他什么鬼的，随它去吧，反正我不造。

Git 从哪里来，到哪里去？

My name is Linus, and I am your God.

有些人就是这么叼，敢说出这句话，而且你还不能不服。

就是右边这个，见到他真人记得膜拜下。

为了开发神器 Linux ，他又创造了一大神器，Git ，就酱。

到哪里去？有代码在的地方，就有Git.



Git 入门

Hello World

用世界上最大的同性交友网站作为展示。

`https://github.com/git/git`

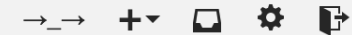
`https://github.com/torvalds/linux`

具体其他例如 Bitbucket、GitCafe、Gitlab，都类似



This repository Search

Explore Gist Blog Help



Watch 936

Star 8,334

Fork 4,766

Git Source Code Mirror - This is a publish-only repository and all pull requests are ignored. Please follow Documentation/SubmittingPatches procedure for any of your improvements.

38,898 commits

5 branches

542 releases

870 contributors



branch: master

git / +



Post 2.3 cycle (batch #8)



gitster authored 3 days ago

latest commit d67f9d5e8f

Documentation

Merge branch 'ms/submodule-update-config-doc'

3 days ago

block-sha1

Merge branch 'jk/pack-bitmap'

a year ago

builtin

Merge branch 'ja/clean-confirm-i18n'

3 days ago

compat

Merge branch 'dm/compat-s-ifmt-for-zos'

3 months ago

contrib

Merge branch 'av/wincred-with-at-in-username-fix'

19 days ago

ewah

Merge branch 'jk/pack-bitmap' into maint

4 days ago

git-gui

Merge git://repo.or.cz/git-gui

9 months ago

gitk-git

Merge git://ozlabs.org/~paulus/gitk

4 months ago

gitweb

gitweb: hack around CGI's list-context param() handling

4 months ago

<> Code

Pull Requests 17

Pulse

Graphs

SSH clone URL

git@github.com:git

You can clone with [HTTPS](#), [SSH](#), or [Subversion](#).

Clone in Desktop

Download ZIP



This repository Search

Explore Gist Blog Help



torvalds / linux

Watch

3,226

★ Unstar

20,148

Fork

7,989

Linux kernel source tree

506,297 commits

1 branch

409 releases

4,742 contributors



branch: master

linux / +



Merge git://git.kernel.org/pub/scm/virt/kvm/kvm



torvalds authored an hour ago

latest commit affb8172de

Documentation	Merge git://git.kernel.org/pub/scm/linux/kernel/git/davem/net	2 hours ago
arch	Merge git://git.kernel.org/pub/scm/virt/kvm/kvm	an hour ago
block	blk-throttle: check stats_cpu before reading it from sysfs	17 days ago
crypto	Merge git://git.kernel.org/pub/scm/linux/kernel/git/herbert/crypto-2.6	23 days ago
drivers	Merge branch 'for-linus' of git://git.kernel.org/pub/scm/linux/kernel...	2 hours ago
firmware	kbuild: remove obj-n and lib-n handling	5 months ago
fs	Merge branch 'for-linus' of git://git.kernel.org/pub/scm/linux/kernel...	3 days ago
include	Merge git://git.kernel.org/pub/scm/linux/kernel/git/davem/net	2 hours ago
init	Merge branch 'kconfig' of git://git.kernel.org/pub/scm/linux/kernel/g...	18 days ago

<> Code

Pull Requests 67

Pulse

Graphs

SSH clone URL

git@github.com:tor



You can clone with [HTTPS](#), [SSH](#), or [Subversion](#).

Clone in Desktop

Download ZIP

Hello World

找到并复制：

```
git@github.com:git/git.git
```

```
git@github.com:torvalds/linux.git
```

这样子的链接， `git@domain:(user|group)/repo.git` ， 通过 `ssh` 方式进行 `clone` 。

强烈不推荐通过 `https` 方式克隆， 没啥原因， 就是每次远程交互都要填写密码而已。

指令：

```
git clone git@github.com:torvalds/linux.git
```

， 然后你就看到当前目录有一个 `linux` 的目录了。

算是入门了。当然， 想用 `ssh` 方式 `clone` ， 需要先去网站提交自己的 `ssh key` 。 因此， 使用 `*nix` 杠杠的无压力。

Git 常用指令

入门指令（GUI阶段）

clone	checkout	push	pull	add	commit

进阶指令（命令行）

remote	fetch	rebase	diff	push -f	reset
add	branch	status	log	stash	tag
merge	bisect				

其他的没提到的，基本上在 ZoomEye Git 协助实践中没用到。

更多：<http://git-scm.com/docs>

ps:我乱排的，仅代表个人意见。

Git 团队协作方式

如何使用 Git 进行团队协作？

每个团队都有适合自己的协助模式。多种多样。

多种多样的协助模式，有些团队禁止 fork ，有些团队禁止 merge 。

找到适合自己团队的模式。

ZoomEye 团队协作思想是：历史不可改变。主分支禁止 push。

其他团队有自己不同的协助流程，例如：

唱吧客户端团队的Git实践：<http://www.iwangke.me/2015/02/08/Git-in-action-at-ChangBa/>

ZoomEye 团队协作思想是：历史不可改变。主分支禁止 push。（我胡扯出来的）

处事掌握了思想，就能找到方法论。

那如何使用该思想进行指导团队协作。

历史不可改变，主分支禁止 push，内涵是什么？

对团队进行影响的所有操作，都属于历史，不可改变。
在 git 里面，就是主开发分支不可进行历史变动。

然后就延伸出主分支禁止任何 reset 操作，
有 reset 必有 push ，这里只允许 git 服务器程序进行主分支 push ，禁止个人 push

打个针： 一般主分支dev，主发布分支为 master，主仓库基本上两个分支即可。

ps： 仓库初始化时候，不受上述的影响，初始化都是一个人进行，多人的时候遵从这个模式的阶段。

如何做到：历史不可改变，主分支禁止 push

引申出：

- 1、任何情况下，主分支 log 记录不变。每一次合并到主分支的功能都进行测试，
确保经过模拟线上测试，并且没有已知的bug。
 - 2、主分支只能通过 mr accept 方式进行功能的合并。
- 因此，就产生了 ZoomEye 的协作 Git 合作流程

ZoomEye 团队 Git 协作流程

1、git 组织架构：存在一个 group ， 成员加入到 group 中，同时设定好对应权限。
设定主开发分支为 dev ， 主生产分支为 master

2、协助流程：

1) 每一个成员，参与每一个项目，都需要进行 fork repo 操作。开发时候自由创建分支。

2) 进行功能开发完毕之后，提交到 origin ， 创建 mr 到 upstream ， assign 给非自己的团队成员进行功能审核和 review ， 审核阶段需要回应每一个 comment 。

3) 每个被 assign 的成员，在觉得 review 已经修复完成之后，觉得可以合并到主分支，就写一个 :+1: （大拇指）或者 lgtn ， 代表自己觉得这个 mr 可以合并了。

4) 确保 mr 是有一个 Accept Merge Request 的按钮，而不是 This request can't be merged with GitLab. 的提示。这个时候就需要进行 rebase 操作了。

5) 如何进行 rebase ？ 一会讲。这个是 ZoomEye 团队协助流程的灵魂。

3、发布流程：

1) 主分支用于发布测试，另外有个生产环境的分支。

2) 每一次提交 mr ， 都需要进行 mr 测试。

3) mr 测试成功，合并到主分支，发布正式测试环境。

4) 正式测试成功，找个良辰吉日，提交 mr 到 发布分支，选择发布生产环境。

测试展示 MR

From


whoami:→_→

 into

group:→_→

Download as ▾

Accept Merge Request

 **Modify commit message**

If you still want to merge this request manually - use [command line](#)

-  Discussion

2
-  **Commits**

2
-  Changes

2

 10 Mar, 2015 2 commits	<div>234567i6543224567897654324567 测试 commit ...</div> <div>→_→</div>	Browse Code » about 14 hours ago
	<div>sadfgjhkh 测试 commit ~~~</div> <div>→_→</div>	Browse Code » about 15 hours ago

Open

Merge Request #10086 · created by [whoami](#) 7 days ago

Close

测试展示 mr

From [whoami](#):→_→ into [group](#):→_→

Download

This request can't be merged with GitLab.

You should do it manually with [command line](#)

Discussion 22

Commits 11

Changes 30

rebase 是什么鬼？

官方解析：Forward-port local commits to the updated upstream head

```
git rebase [-i | --interactive] [options] [--exec <cmd>] [--onto <newbase>]  
[<upstream> [<branch>]]
```

```
git rebase [-i | --interactive] [options] [--exec <cmd>] [--onto <newbase>] --root  
[<branch>]
```

我们常用的：

```
git rebase --continue | --skip | --abort | --edit-todo
```

```
git rebase -i upstream/dev
```

```
git rebase --continue
```

```
git rebase --abort
```

```
git rebase --skip
```

屁话太多，来实战？

先来个故事，具体故事可查看：<http://shenqihui.github.io/2015/01/29/best-practices-of-git/>

这里是精简版的故事。

顺路讲解下怎么协助。

背景：四个人，开发个 谷猫 。git 网站地址：git.zoomeye.org

一个 Git group 叫做 goocat，创建一个 goocat 的 repo，主开发分支 dev，主生产分支 master。主仓库已经在成熟开发阶段，各个人员都在有序的进行功能开发。

git id:

小明： ming

小花： hua

小猪： zhu

小胖： pang

场景：

小明和小花协助开发个用户头像上传的功能，下称 A 功能。

小猪和小胖协助开发个用户密码相关的功能，下称 B 功能。

需求已经订好，产品经理靠边站。

A 功能正在开发，小明先行把架构搭好，小花实现交互前端功能。
小明搭好框架，告知小花可以在这个基础上开发前端功能了。
小明提交到自己的 origin 仓库的 user-img 分支，并告知小花，
小花的操作：

git remote -v # 看看自己的远程仓库，只有 origin 和 upstream，于是增加远程仓库

➔ goocat git:(user-img) git remote -v	
origin	git@git.zoomeye.org:hua/goocat.git (fetch)
origin	git@git.zoomeye.org:hua/goocat.git (push)
upstream	git@git.zoomeye.org:goocat/goocat.git (push)
upstream	git@git.zoomeye.org:goocat/goocat.git (fetch)

git remote add ming git@git.zoomeye.org:ming/goocat.git # 增加小明的远程仓库
git fetch ming # 获取小明的最新代码
git rebase -i ming/user-img # 重置自己的本地仓库。
不断 coding 增加功能。提交、推送。前端基本开发完成。交给小明整合接口。
git push -f origin user-img

同样，小明也要更新小花最新的代码进行功能增加。

```
git remote add hua git@git.zoomeye.org # 增加小花的远程仓库
```

```
git fetch 花# 获取小花的最新代码
```

```
git rebase -i hua/user-img # 重置自己的本地仓库。
```

不断 coding 增加功能。提交、推送。前端基本开发完成。交给小明整合接口。

```
git push -f origin user-img
```

测试完成了。功能开发成功。

应该创建 mr 了。

You pushed to **user-img** at 小明 / **goocat** 19 minutes ago

Create Merge Request

点击按钮，创建 mr ，写好理由

New merge request

From `ming:ics` into `goocat:dev`

[Change branch](#)

Title *

用户头像上传

Description

Write

Preview

[↗ Edit in full](#)

1. 用户 model
2. 用户 view
3. 用户 template

Parsed with [Gitlab Flavored Markdown](#).

Attach images (JPG, PNG, GIF) by dragging & dropping or [selecting](#)

 Assign to

小花

[Assign to me](#)

 Milestone

Select milestone

[Create new milestone](#)

 Labels

[Create new label](#)

Submit merge request

 Commits 7

 Changes 27

Merge request was successfully created.

Open

Merge Request #10087 · created by 小明 about a minute ago

Close

用户头像上传

1. 用户 model
2. 用户 view
3. 用户 template

From xiaoming:user-img into goocat:dev



This request can't be merged with GitLab.

You should do it manually with [command line](#)

Discussion 0

Commits 7

Changes 27

只可惜，出现了：

This request can't be merged with GitLab.
You should do it manually with **command line**

不要怕，有 rebase 在。

如何使用 rebase 。

1、先进行当前分支的备份，以防 rebase 出了问题就~~~~了。

备份方式多种多样，checkout 新分支 / push 本分支最新代码到远程仓库。

2、更新代码：

git fetch upstream # 获取最新代码

git rebase -i upstream/dev # 重置仓库 commit 头。

解决冲突

git rebase --continue | --abort | --skip # rebase 中途的很多操作。

3、rebase 完毕，检查功能是否没错误。没错误就 push 到远程；有错误就思考下刚刚为啥错了，然后再从备份的地方拿回代码（一般很少出现，认真 rebase 即可。）

4、打开网页，看到类似下图的按钮，就 rebase 完成。

5、说明一点，多个 mr 合并了，估计还是有问题，因此，可能需要多次 rebase 的。

Open

Merge Request #10087 · created by 小明 15 minutes ago

Close

Edit

用户头像上传（用来测试的。！！！！！！别动。）

- 1. 用户 model
- 2. 用户 view
- 3. 用户 template

From 小明:user-img into goocat:dev

Download as

Accept Merge Request

Modify commit message

If you still want to merge this request manually - use [command line](#)

Discussion 5

Commits 4

Changes 17

Showing 17 changed files

Show diff stats

Inline

Side-by-side

说说你自己的坑

为啥这样做：

- 1、时刻保持 zoomeye.org 能用
dev 主开发分支， master 主生产分支，确保 master 可用。
- 2、确保代码质量
只通过 server 的 mr 自动合并代码，每个 mr 进行 review
同事提高成员的代码水平
- 3、折中高效的协助方式
通过 fetch 穿插每一个成员的每一个 commit
- 4、确保测试通过
mr 测试，合并前测试
dev 测试，合并后测试
master，发布生产

部分链接

git分支最佳实践: <http://segmentfault.net/a/11900000000434973>

git 魔法: http://www-cs-students.stanford.edu/~blynn/gitmagic/intl/zh_tw/ch05.html

廖雪峰写的多人协作文章（自行谷歌）

沉浸式学 Git: <http://igit.linuxtoy.org/index.html>

一个故事: <http://shenqihui.github.io/2015/01/29/best-practices-of-git/>

唱吧 git 协助流程: <http://www.iwangke.me/2015/02/08/Git-in-action-at-ChangBa/>

git 历史 : <https://lkml.org/lkml/2005/4/6/121>

王垠的 git 感想: <http://www.yinwang.org/blog-cn/2015/03/11/git-etiquette/>



@知道创宇 ZoomEye Team