

# git & github版本控制及代码仓库托管

---团队开发方法

## A 阶段：最基本操作（基础部分，上午）

注册github账号 登录：<https://github.com/>

A-1: 建立本地仓库--初始化本地仓库

```
$ git init //在本地的目录建立本地仓库
```

A-2：建立自己的git专属名称及邮箱

```
$ git config --global user.name "your name" //名字不要用中文
```

```
$ git config --global user.email "your_email@example.com" //填写自己的邮件地址
```

A-2：建立自己SSH key密码操作

```
$ ssh-keygen -t rsa -C "your_email@example.com" //填写自己的邮件地址
```

A-3: Fork仓库（注意区分fork与clone的区别）

github.com/RT-Thread/rt-thread

Gmail YouTube 地图 开课吧学习中心 码云 Gitee + Jenki... 我的网易云课堂 利用gitee构建jenki... CI持续集成系统环... yangyangwithgnu... 通过C语言项目基于...

Search or jump to...

Pull requests Issues Marketplace Explore

RT-Thread / rt-thread

Watch

515

Star

4.1k

Fork

2.6k

Code

Issues 192

Pull requests 22

Actions

Projects 1

Wiki

Security

Insights



Search or jump to...

Pull requests Issues Marketplace Explore

Set status

lloodao

lloodao

Edit profile

lloodao@163.com

Overview

Repositories 25

Projects 0

Packages 0

Stars 36

Followers 0

Following 4

Popular repositories

gitkills

★ 1

rt-thread

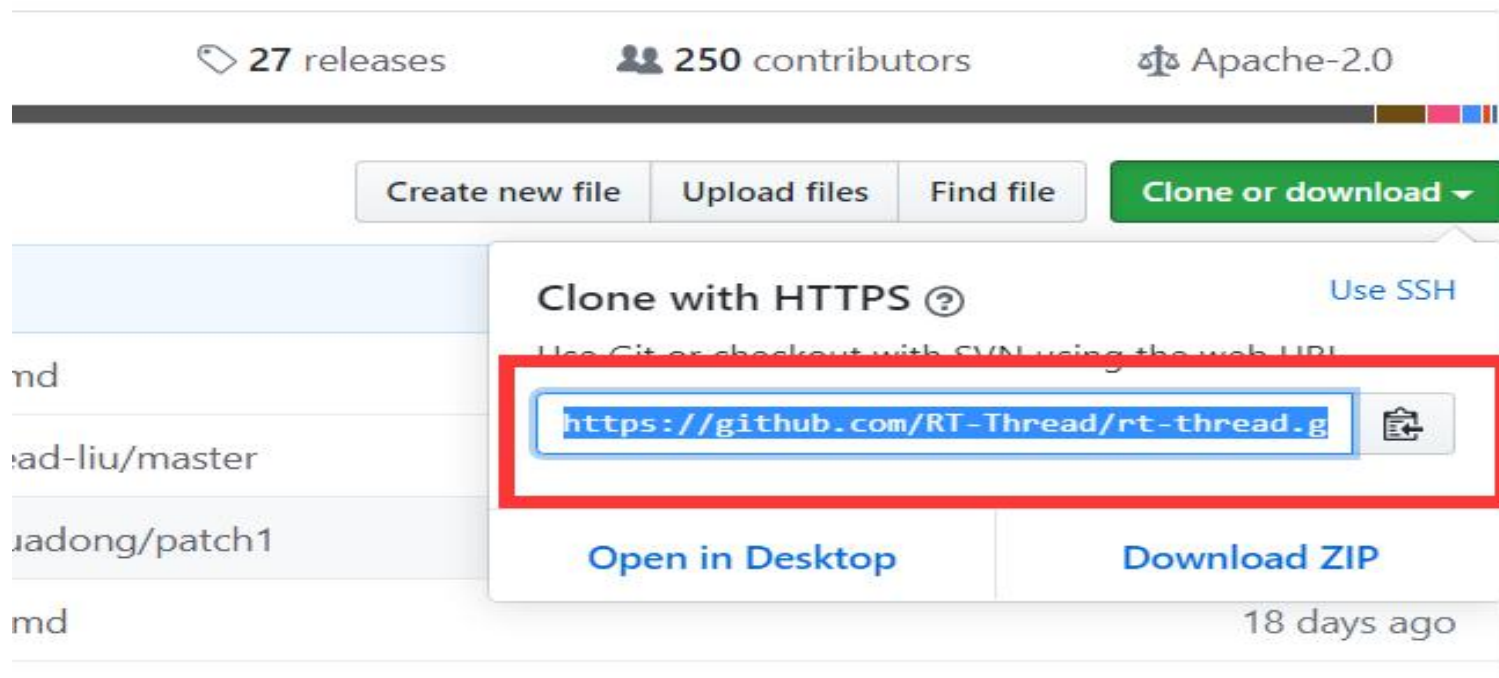
Forked from RT-Thread/rt-thread

RT-Thread is an open source IoT operating system from China.

● C ★ 1

A-4: clone托管仓库中的项目代码到本地（可以clone自己或别人的托管仓库中的项目），从这一步骤开始进入团队协作的方式，克隆到本地之后可以改写，添加，编译，测试代码；

```
$ git clone https://github.com/RT-Thread/rt-thread //克隆RT-Thread的项目到本地硬盘
```



```
E:\rt-thread-v4.0.1\rt-thread>dir
```

驱动器 E 中的卷是 Data

卷的序列号是 80B3-FA7F

E:\rt-thread-v4.0.1\rt-thread 的目录

```
2019/05/16  20:25    <DIR>          .
2019/05/16  20:25    <DIR>          ..
2019/05/16  20:25                459 .gitattributes
2019/05/16  20:25    <DIR>          .github
2019/05/16  20:25                268 .gitignore
2019/05/16  20:25            8,191 .travis.yml
2019/05/16  20:25                571 AUTHORS
2019/05/16  20:25    <DIR>          bsp
2019/05/16  20:25        69,913 ChangeLog.md
2019/05/16  20:25    <DIR>          components
2019/05/16  20:25    <DIR>          documentation
2019/05/16  20:25    <DIR>          examples
2019/05/16  20:25    <DIR>          include
2019/05/16  20:25                100 Kconfig
2019/05/16  20:25    <DIR>          libcpu
2019/05/16  20:25        11,357 LICENSE
2019/05/16  20:25            4,462 README.md
2019/05/16  20:25            5,939 README_zh.md
2019/05/16  20:25    <DIR>          src
2019/05/16  20:25    <DIR>          tools
                9 个文件          101,260 字节
                11 个目录    5,258,182,656 可用字节
```

```
E:\rt-thread-v4.0.1\rt-thread>_
```

## B 阶段：本地仓库的各项操作

B-1： 在克隆下来的项目目录任何改变都在git的监管之下（虽然还没有与本地库与托管库关联），下面这个指令就可以查阅所有变化的

```
$ git status          //查阅在本地库目录中的变化
```

B-2： 提交到暂存区

```
$ git add . (file1,file2...) //指令之后空格加点号（所有改变的文件）或者指定以改变的文件
```

B-3： 提交日志及查阅日志

```
$ git commit -m "本次提交的日志概述"
```

```
$ git log (-p)          //查看提交的日志概述, -p只查阅改变
```

```
$ git diff              //查阅更改前后的差别
```

```
$ git diff HEAD         //查看工作树和最新提交的差别，养成习惯，执行commit前执行下这条指令
```

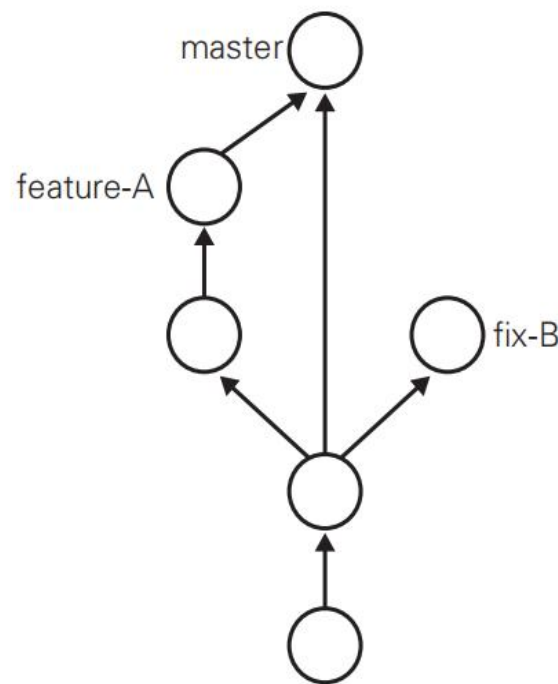
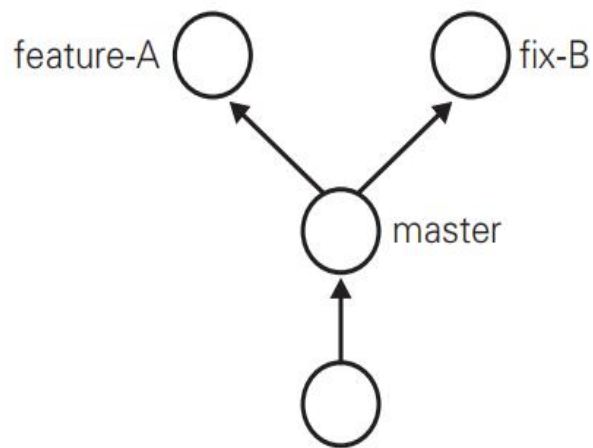
B-4: 提交到托管仓库，以上的操作都是作用于暂存区（属于本地范畴），执行到这一步，本地的暂存区就开始与托管仓库关联起来了

```
$ git push    //把本地所有变更推送到托管仓库，如在托管仓库还没有添加ssh密钥，会提示要求输入用户名及邮件
```

以上A，B的一轮操作，全新的代码开发模式开始打开了一扇门，请热情的投入它的怀抱吧，你现在已经同步全球的代码开发模式，地球最牛B的大拿你都可以骚扰（我偶尔会给IBM的开源大牛吐槽，issues linux的父亲linus）

## git的进阶操作

多个并行操作就会用到分支，多人协同会并存多个最新的代码；从master分支之后，每个分支都拥有各自的最新代码。master分支是git默认创建的分支，master分支成为主干分，因此基本所有开发以这个分支为中心进行，各分支完成之后就可以合并到master分支。






## 分支操作

\$ git branch

//查阅当前的分支，\*号就是当前的处于的分支

 命令提示符

```
E:\project_repo\helloworld>git branch
```

```
* master
```

```
E:\project_repo\helloworld>
```

\$ git checkout -b train-branch //建立一个名为train-branch的特征分支

 命令提示符

```
E:\project_repo\helloworld>git branch
```

```
* master
```

```
E:\project_repo\helloworld>git checkout -b train-branch
```

```
Switched to a new branch 'train-branch'
```

```
E:\project_repo\helloworld>
```

C:\> 命令提示符

```
E:\project_repo\helloworld>git branch
```

```
* master
```

```
E:\project_repo\helloworld>git checkout -b train-branch
```

```
Switched to a new branch 'train-branch'
```

```
E:\project_repo\helloworld>git branch
```

```
master
```

```
* train-branch
```

```
E:\project_repo\helloworld>
```

## 切换分支操作

\$ git checkout master //切换到master分支

\$git checkout - //切换回到上一个分支

命令提示符

```
E:\project_repo\helloworld>git branch
```

```
* master
```

```
E:\project_repo\helloworld>git checkout -b train-branch
```

```
Switched to a new branch 'train-branch'
```

```
E:\project_repo\helloworld>git branch
```

```
master
```

```
* train-branch
```

```
E:\project_repo\helloworld>
```

把本地分支推送到远程仓库

```
git push --set-upstream origin train-branch
```

lloodao / helloworld

Unwatch 1

Unstar 1

Fork 1

<> Code

Issues 0

Pull requests 0

Actions

Projects 0

Wiki

Security

Insights

Settings

Overview

Yours

Active

Stale

All branches

Search branches...

Default branch

master Updated yesterday by hunkhand

✓

Default

Change default branch

Your branches

train-branch Updated 13 minutes ago by hunkhand

✓

0 | 1

New pull request

Active branches

train-branch Updated 13 minutes ago by hunkhand

✓

0 | 1

New pull request

打开github查看这个指令操作之后的变化，在train-branch分支中多了一个文件train-branch

27 commits

2 branches

0 packages

0 releases

2 contributors

Your recently pushed branches:

🔗 train-branch (18 minutes ago)

🔗 Compare & pull request

Branch: train-branch ▾

New pull request


Create new file

Upload files

Find file

Clone or download ▾

This branch is 1 commit ahead of master. 🔗 Pull request 📄 Compare

 hunkhand	add a file of train-branch to tran-branch	✓ Latest commit ef1455b 19 minutes ago
📁 course_2_2	add yml to course_2_2	15 days ago
📁 documents	add new file code CD/CD & modified readme.md	10 days ago
📄 .travis.yml	modified .travis.yml	yesterday
📄 .travis.yml~	modified .travis.yml	yesterday
📄 README.md	add new file code CD/CD & modified readme.md	10 days ago
📄 hello.c	make documents floder && doxygen manual	15 days ago
<div>📄 train-branch</div>	add a file of train-branch to tran-branch	19 minutes ago

# 合并分支

合并分支的步骤：1) 首先切换到master分支；2: 提交合并


```
$ git checkout master //切换到master分支
```

```
$ git merge --no-ff train-branch //合并train-branch分支
```

C:\A. 命令提示符

```
E:\project_repo\helloworld>git branch
* master
  train-branch

E:\project_repo\helloworld>git merge --no-ff train-branch
Merge made by the 'recursive' strategy.
  train-branch | 1 +
  1 file changed, 1 insertion(+)
  create mode 100644 train-branch
```



```
E:\project_repo\helloworld>
```



\$git log--graph //也可以显示哈希码

命令提示符 - git log --graph

```
create mode 100644 train-branch

E:\project_repo\helloworld>git log --graph
*   commit ffc11a6abdf78377ae6da9dbd8f34bd8abf6f9c6 (HEAD -> master)
   \
    Merge: 5667755 efl455b
    Author: loodao <hunkhand@163.com>
    Date:   Thu Mar 5 15:51:18 2020 +0800

        Merge branch 'train-branch'

*   commit efl455b07aa7d1bcd62f6a633937c011fdf21cc8 (origin/train-branch, train-branch)
   /
    Author: loodao <hunkhand@163.com>
    Date:   Thu Mar 5 15:19:57 2020 +0800

        add a file of train-branch to tran-branch

*   commit 56677551c210199b09f86d1abd52c235c8fe3d51 (origin/master, origin/HEAD)
   /
    Author: loodao <hunkhand@163.com>
    Date:   Wed Mar 4 14:01:12 2020 +0800

        modified .travis.yml

*   commit fb62e53f33c43f43818bae72dfe4be06acad4e82
   /
    Author: loodao <hunkhand@163.com>
    Date:   Mon Feb 24 14:26:46 2020 +0800

        add new file code CD/CD & modified readme.md

*   commit f95780386c7df44192f2a864cf3a518770f4aeb1
   /
    Author: loodao <hunkhand@163.com>
    Date:   Mon Feb 24 12:08:00 2020 +0800

        :

```



## 有后悔药可吃，时间隧道玩穿越---git可以做到

回溯历史版本，世界真的可以重来


\$ git reset --hard (哈希值) //这里的哈希值是

tips: 查看每次提交时段的哈希值 \$git log--graph

命令提示符

```
E:\project_repo\helloworld>git rev-parse
```

```
E:\project_repo\helloworld>git rev-parse HEAD  
ffc11a6abdf78377ae6da9dbd8f34bd8abf6f9c6
```



```
E:\project_repo\helloworld>
```

选择命令提示符 - git log --graph

The most similar commands are  
branch  
grep

E:\project\_repo\helloworld>git log

commit 17799800bfcb857653746ff1c6faaaf5e44ad1c0 (HEAD -> my\_code, origin/my\_code, master)

Merge: 5667755 ef1455b

Author: loodao <hunkhand@163.com>

Date: Thu Mar 5 15:51:18 2020 +0800

Merge branch 'train-branch'

commit ef1455b07aa7d1bcd62f6a633937c011fdf21cc8 (origin/train-branch, train-branch)

Author: loodao <hunkhand@163.com>

Date: Thu Mar 5 15:19:57 2020 +0800

add a file of train-branch to tran-branch

commit 56677551c210199b09f86d1abd52c235c8fe3d51 (origin/master, origin/HEAD)

Author: loodao <hunkhand@163.com>

Date: Wed Mar 4 14:01:12 2020 +0800

modified .travis.yml

commit fb62e53f33c43f43818bae72dfe4be06acad4e82

Author: loodao <hunkhand@163.com>

Date: Mon Feb 24 14:26:46 2020 +0800

哈希值



## 永远保持最新的代码

因为多人协助开发，多个分支时刻有完成的代码推送，为了保持自己本地仓库的项目代码与远程仓库保持更新应该养成定时从远程仓库拉最新的代码

```
$ git pull          //从远程仓库获取最新的代码
```

到此基本的协同开发基本操作已经足够！

小结： 推荐更好的学习git的网站链接： <https://learngitbranching.js.org> ； <http://try.github.io>

官网的更需要看了，现在有中文的说明了： <https://git-scm.com/book/zh/v2>

git-scm.com/book/zh/v2

3mailYouTube地图开课吧学习中心码云 Gitee + Jenki...我的网易云课堂利用gitee构建jenki...CI持续集成系统环...yangyangwithgnu...通过C译

git

--everything-is-local

About

Documentation

Reference

Book

Videos

External Links

Downloads

Community

This book is available in [English](#).

Full translation available in

[български език](#),

[Español](#),

[Français](#),

[Ελληνικά](#),

[日本語](#).

Book

The entire Pro Git book, written by Scott Chacon and Ben Straub and published by Apress, is available here. All content is licensed under the [Creative Commons Attribution Non Commercial Share Alike 3.0 license](#). Print versions of the book are available on [Amazon.com](#).

1. 起步

1.1 关于版本控制

1.2 Git 简史

1.3 Git 基础

1.4 命令行

1.5 安装 Git

1.6 初次运行 Git 前的配置

1.7 获取帮助

1.8 总结

Pro Git

2nd Edition (2014)

Download Ebook

pdf

epub

amazon

Learn Git Branching

learn git branching.js.org

应用GmailYouTube地图开课吧学习中心码云 Gitee + Jenki...我的网易云课堂利用gitee构建jenki...其他书签

学习 Git 分支

\$ help

\$ levels

欢迎光临 Learn Git Branching

你对 Git 感兴趣吗？那么算是来对地方了！“Learning Git Branching”可以说是目前为止最好的教程了，在沙盒里你能执行相应的命令，还能看到每个命令的执行情况；通过一系列刺激的关卡挑战，逐步深入的学习 Git 的强大功能，在这个过程中你可能还会发现一些有意思的事情。

关闭这个对话框以后，你会看到我们提供的许多关卡。如果你是初学者，从第一关开始逐个向后挑战就是了。而如果你已经入门了，可以略过前面，直接挑战后面更有难度的关卡。

演示

如果你还没看过演示，请[到此](#)查看。

PS：想直接进入沙盒？在 URL 后头加上 `?NODEMO` 就可以了，试一下[这个链接](#)：

Fork me on GitHub

## 高阶操作github/gitlab/gitee等（下午）

上编已经简单使用了克隆远程仓库指令`git clone git.XXXXX`；下面介绍对github的高级操作，这是对代码推送，合并，审核，版本发放的操作

A-1：快速提高自身代码质量，让自己早日也成为“不服，代码见”！

阅读大拿们的神级代码了；当你发现某大牛或某公司的开源代码对自己非常有用，抑或参与某一开源项目，再就是关注某一项目的进展，你就要学会如下操作。

登录大牛的GitHub链接，点击上面的follow（显示别人关注你的）；watch是你关注别人的，比如我一直关注RT-Thread开源项目的更新，就在他们的项目上面的watch点击，那么我就时刻知道他们这个项目最新的代码更新及进展了。

## 傍大牛的项目，关注项目代码进展

github.com/RT-Thread/rt-thread

Gmail YouTube 地图 开课吧学习中心 码云 Gitee + Jenki... 我的网易云课堂 利用gitee构建jenki... CI持续集成系统环... yangyangwithgnu... 通过C语言项目基于...

ch or jump to... Pull requests Issues Marketplace Explore

RT-Thread / rt-thread

Watch 515 Star 4.1k Fork 2.6k

<> Code Issues 192 Pull requests 22 Actions Projects 1 Wiki Security Insights

RT-Thread is an open source IoT operating system from China. <http://www.rt-thread.org>

## A-2: Pull Requests

提交PR分享代码给团队，团队协作开发就是以PR为基础的

提交的代码已经通过项目创建者或授权审核人员Code Review (审核过)。这部分代码是这个项目精华，多读得益匪浅。也可学到代码审核人员对代码审核的具体要求，如你提供的代码通过项目审核组人员认可，恭喜你，假以时日你就是大牛了（contributions贡献者）！呵呵 ....

## A-3: issues

提问题，在这里可以学到太多东西，连别人的提问问题的方式及问题关键点都全部呈现（tips:学外语的一个手段之一，）可以学到很多世界各地关于这个项目的意见及吐槽，项目管理者的尖酸刻薄，哈哈！


下面这句话摘自网络

“Issues（问题单）是一种伟大的工作方式，它用于对项目进行跟踪、增强和排错。它们就像电子邮件一样——除了它们可以与团队的其他成员进行分享和讨论。大多数软件项目都会有某种错误跟踪器，GitHub 的错误跟踪器称为“Issues”，并且在每个仓库中都有自己的 Issues 部分。



下面是我写这个文稿的时候，当时RT-Thread的issues数量及提出issues的部分问题

[Code](#) **Issues 192** [Pull requests 22](#) [Actions](#) [Projects 1](#) [Wiki](#) [Security](#) [Insights](#)

 **Want to contribute to RT-Thread/rt-thread?** [Dismiss](#)

If you have a bug or an idea, browse the open issues before opening a new one. You can also take a look at the [Open Source Guide](#).

Pinned issues

**中文帖可以发到Gitee或RT-Thread中文论坛。Here you can place issue in English.**  
#2817 opened on 28 Jun 2019 by BernardXiong  
[Open](#)

[Filters](#)  [Labels 10](#) [Milestones 4](#) [New issue](#)

[192 Open](#) [302 Closed](#) [Author](#) [Label](#) [Projects](#) [Milestones](#) [Assignee](#) [Sort](#)

[Open](#) console烧写完或者软重启不能用  
#3429 opened 5 hours ago by SmallSmartMouse 3

[Open](#) 写文件的问题  
#3428 opened 5 hours ago by SmallSmartMouse

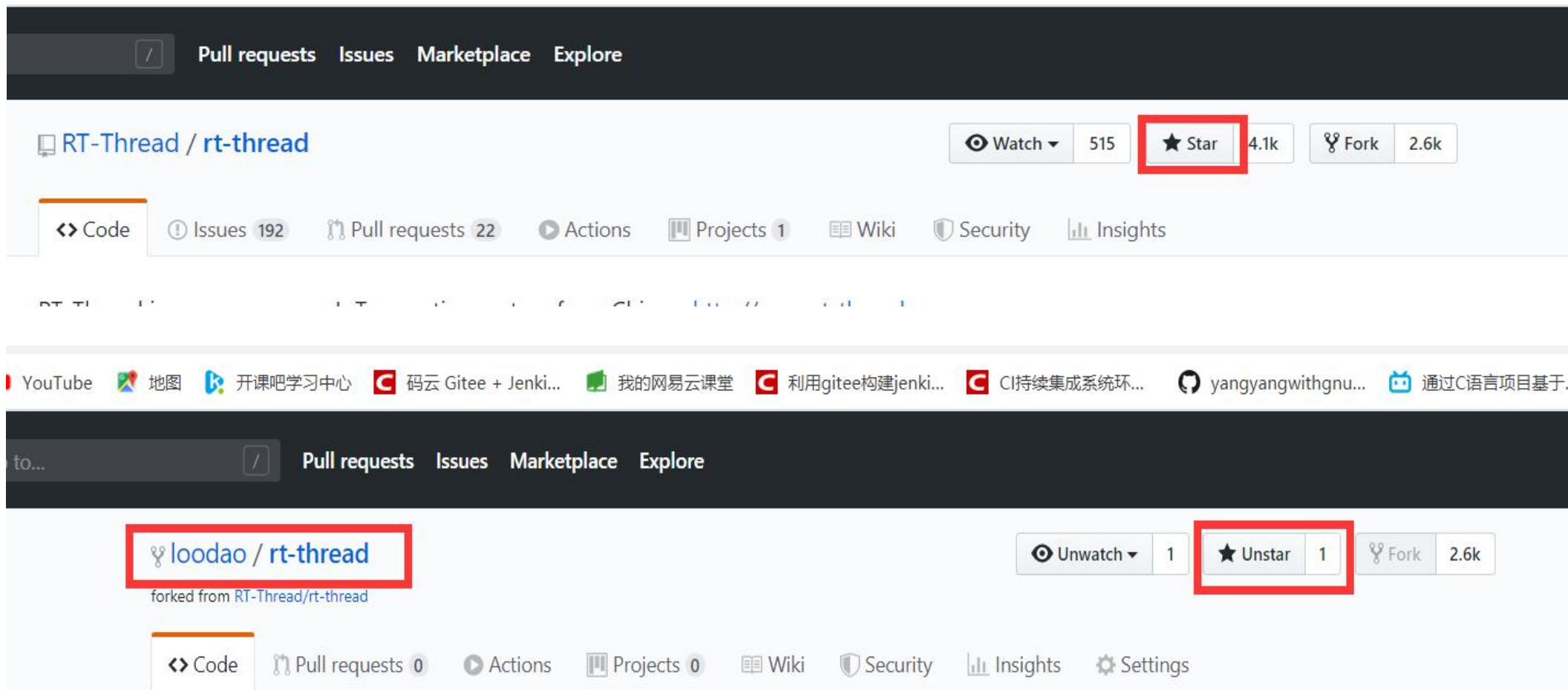
[Open](#) tag 4.02, v4.0.2/bsp/stm32/libraries/HAL\_Drivers/drv\_comm.c, Missing implementation for HAL\_Delay  
#3426 opened 23 hours ago by liulee 1

[Open](#) 有计划适配瑞萨电子的RH850吗? 2



## A-4 : start 标星

这里解释为`关注`或者`点赞`更合适，当你点击 star，表示你喜欢这个项目或者通俗点，可以把理解成朋友圈的点赞吧，表示对这个项目的支持。对某一个项目标星也是为了让别人更容易找到他们。



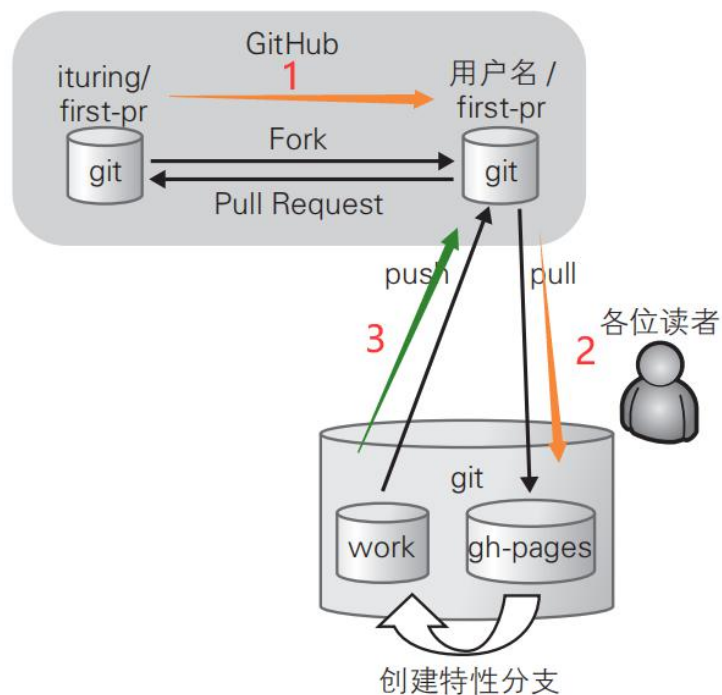
上面这几项github操作就足够丛横各个开源项目，吸取代码清华，提高代码的强壮性

## 如何提交自己的代码到远程仓库

### 1: Pull Request的流程

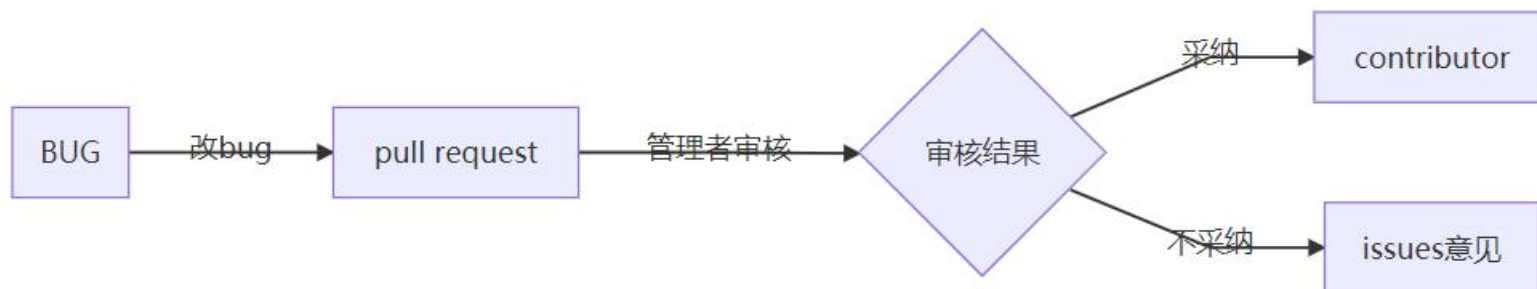
Pull request是自己克隆下来的代码修改之后，请求仓库项目拥有者采纳该修改时采取的一种行为

请看下图的数字表的代码走向



1 fork仓库--》2 clone代码到本地--》4 修改代码--》5 pull到自己远程仓库--》6 pull request

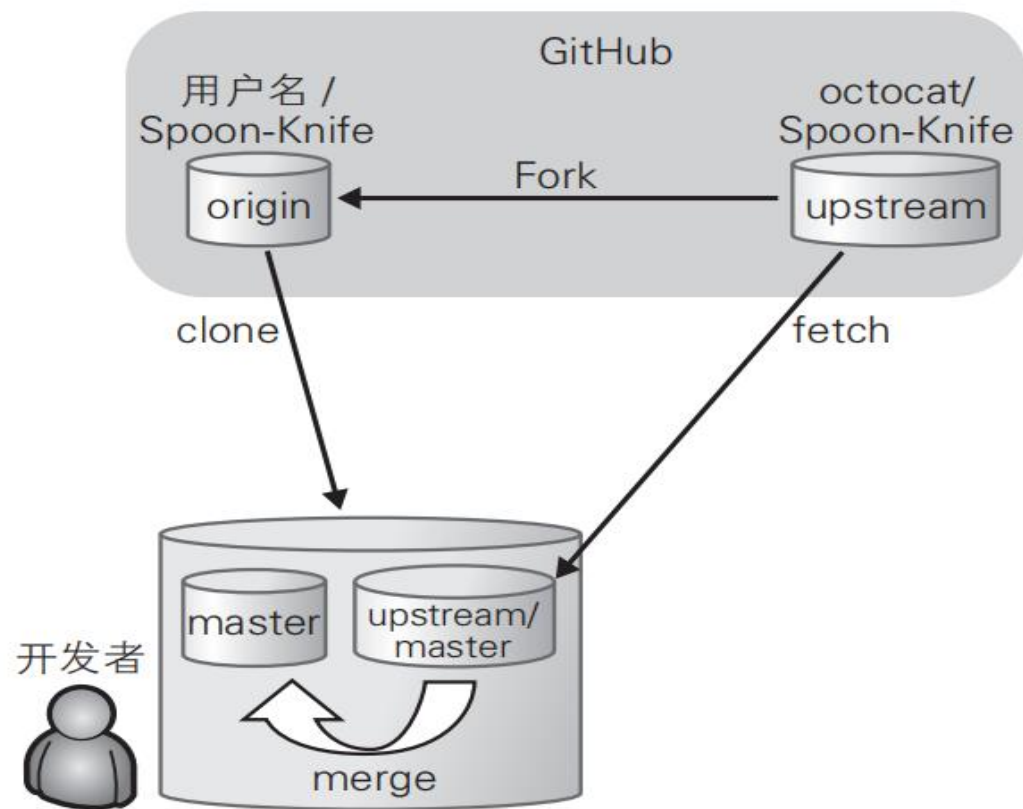
注：其中1，6事要在github上操作



h5 pull request的过程

## 如何让仓库保持最新的状态

clone---> fetch--->merge 让本地仓库保持最新的状态



## 方法1：（在本地操作）

步骤： 1: `$git fetch git@xxxx` //原clonede 仓库

2: `$git remote add upstream git@xxxx` //原clonede 仓库

3: `$git fetch upstream` //保持与原clonede 仓库同步更新

4: `$git checkout -b xxxx` //建立本地特性分支

## 方法2：（在github上操作）

步骤： 1: 在自己的github上fork主仓库

2: 向主仓库发起新的new pull request

3: 选择compare across forks

4: 让后反向操作，base改成自己的fork， head改成原主仓库

5: 每次改代码前先在本地执行git fetch upstream,

loodao / helloworld

Unwatch

1

Unstar

1

Fork

1

Code

Issues

Pull requests 1

Actions

Projects 0

Wiki

Security

Insights

Settings

1

### Label issues and pull requests for new contributors

Dismiss

Now, GitHub will help potential first-time contributors discover issues labeled with **good first issue**

Filters

is:pr is:open

Labels 9

Milestones 0

New pull request

2

1 Open 1 Closed

Author

Label

Projects

Milestones

Reviews

Assignee

Sort

add a file of train-branch to tran-branch ✓  
#2 opened 5 minutes ago by loodao

**ProTip!** Notify someone on an issue with a mention, like: @loodao.

 hunkhand / helloworld

forked from loodao/helloworld

 Watch ▾

0

 Star

0

 Fork

1

 Code

 Pull requests 0

 Actions

 Projects 0

 Wiki

 Security

 Insights

## Comparing changes

Choose two branches to see what's changed or to start a new pull request. If you need to, you can also [compare across forks](#).

5



base repository: hunkhand/helloworld ▾

base: master ▾



head repository: loodao/helloworld ▾

compare: master ▾

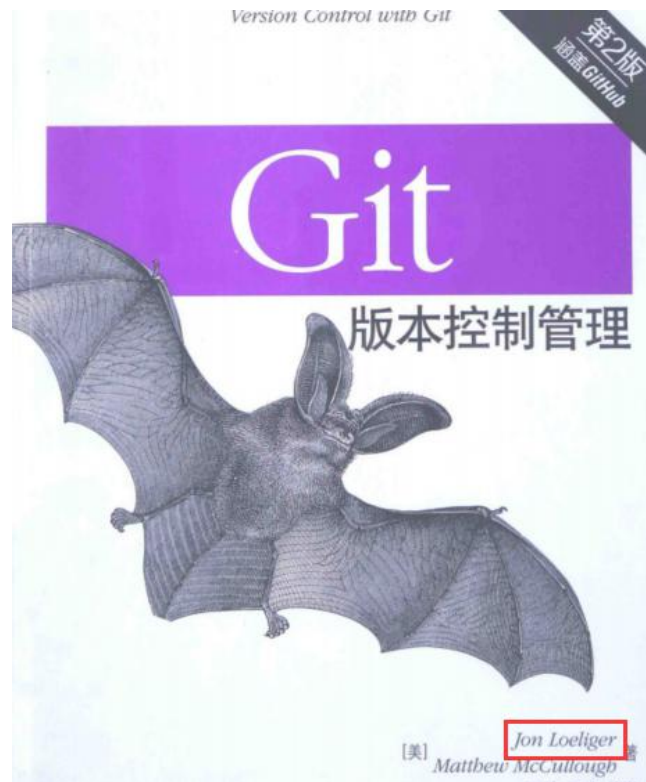
✓ **Able to merge.** These branches can be automatically merged.

# 谢谢

tips: 参考书籍,课后提供



[日] 大塚弘记 / 著  
支鹏浩 刘斌 / 译



[美] Jon Loeliger  
Matthew McCullough 著



# 作业

- 1: 下载git并安装, 建立自己的ssh key密钥
- 2: 注册github账号, fork测试仓库: <https://github.com/loodao/helloworld>
- 3: clone代码之后建立分支
- 4: issues 操作 (提问issues给测试仓库)
- 5: pull request 操作 (修改helloworld.c代码或自己写的代码, 请求PR)
- 6: 设置与主仓库与自己远程仓库的代码更新同步
- 7: 编写代码前的fetch操作
- 8: git push的操作