

GitHub + Travis CI 持续集成

导语

最近在做一个电影售票系统的大作业，这是一个web app，需要前后端分离开发。而且有多人进行协同开发，这就需要使用git这种类型的工具，来帮助我们合并、管理代码。我们选择使用GitHub这个免费的远程代码托管服务网站来进行托管代码。并且使用Travis CI来进行持续集成开发测试。

Git

在之前，也一直使用GitHub，但主要无非用来管理自己平时个人写的代码，比较少涉及到多人协同开发以及解决一些冲突的问题。所以有必要再补一下这方面相关的知识。Git是Linux之父Linus当年在开发linux的过程中，不满于SVN的使用，而自己写出来的一个工具（不愧是大神，实在佩服）。

这里可以参考[廖雪峰Git教程](#)，比较简明详细、清晰明了。主要介绍了如何具体实际来操作Git，适合快速上手。在后续的开发过程中，我也会将此类相关注意点和问题再整理成博客的。

也可以参考[Pro Git](#)这本书，里面比较详细地介绍了各种功能使用，以及原理。

branch 分支

分支是团队开发中，一个重要的工具，使得开发更加安全和稳定。

在具体开发过程中使用 `master` 分支来保管稳定版本的代码，使用 `dev` 分支来管理开发过程中的代码，也可以再新建分支来开发新的特性。等开发完成之后，再将稳定的代码合并到 `master` 分支。

通过 `git checkout dev` 可以切换到 `dev` 分支。

```
$ git checkout dev
Switched to branch 'dev'
```

通过 `git branch` 命令来查看所有分支，当前分支带 `*` 号。

```
$ git branch
* dev
master
```

通过 `git merge` 命令来合并指定分支到当前分支。

```
$ git merge dev
```

合并完dev分支之后就可以删除了

```
$ git branch -d dev
```

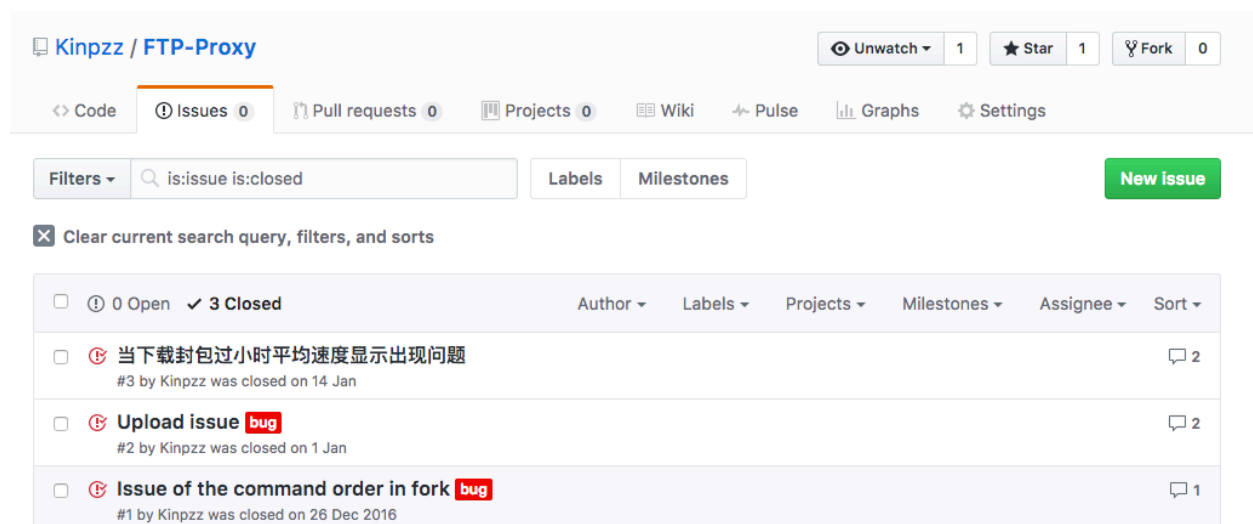
参考: [廖雪峰: 创建于合并分支](#)

GitHub

了解了Git的使用之后, 使用GitHub基本没有什么太大的困难了。开发小组的成员在后续会建立一个GitHub仓库来进行开发。这里推荐使用SSH key来连接到GitHub上, 之后就可以直接使用这个本地的private key来push代码到GitHub的远程仓库上了, 不用再每次push都要输入密码了, 更加安全也方便。官方步骤指南[Connecting to GitHub with SSH](#)

使用Issues进行Bug管理

GitHub提供了一个很好的功能Issues功能, 通过Issues别人可以向项目的开发者提出Bug报告、需求等。我们可以通过使用Issues来记录Bug的发现以及解决。当发现了一个新的Bug之后可以创建一个Issue, 当解决了之后再关闭掉这个Issue。开发者也可以自己向自己的项目提出Issues。下面就先拿我之前的一个仓库使用Issues来进行Bug管理来举例。



Kinpzz / FTP-Proxy

Unwatch 1 Star 1 Fork 0

Code Issues 0 Pull requests 0 Projects 0 Wiki Pulse Graphs Settings

Filters is:issue is:closed Labels Milestones New issue

Clear current search query, filters, and sorts

<input type="checkbox"/>	0 Open 3 Closed	Author	Labels	Projects	Milestones	Assignee	Sort
<input type="checkbox"/>	当下载封包过小时平均速度显示出现问题 #3 by Kinpzz was closed on 14 Jan						2
<input type="checkbox"/>	Upload issue bug #2 by Kinpzz was closed on 1 Jan						2
<input type="checkbox"/>	Issue of the command order in fork bug #1 by Kinpzz was closed on 26 Dec 2016						1

点进具体的某个issue可以发现它是评论列表风格的，可以在里面描述bug的问题，解决之后也可以评论是如何解决这个bug的，可以很方便地记录开发的过程。

Labels 标签功能

并且有一个Labels标签功能，来标记这个Issue是一个Bug还是需求等等。

Upload issue #2

Closed Kinpzz opened this issue on 1 Jan 2017 · 2 comments

Edit New Issue

fork

由于我负责了持续测试这一块，而项目的仓库不是我建立了，在一般的团队协作下，也都是成员fork项目的仓库到自己的帐号下，修改之后再提交pull request。再让自己的项目仓库与上游的项目仓库保持同步，就可以实现我的GitHub帐号来管理Travis了。

先将自己fork的仓库 `git clone "repo_address"` 到本地。

然后通过 `git remote add upstream "upstream_address"`

再通过 `git remote -v` 可以看到多了我们所需同步的上游地址。

```
$ git remote -v
origin    Paul_work:Kinpzz/movie-booking.git (fetch)
origin    Paul_work:Kinpzz/movie-booking.git (push)
upstream  Paul_work:zhzdeng/movie-booking (fetch)
upstream  Paul_work:zhzdeng/movie-booking (push)
```

然后我们只需要保持着 `master` 分支与上游项目仓库保持一致即可，在我们要合并 `dev` 分支到 `master` 分支的时候，先 `git checkout master` 切换到 `master` 分支，再将 `master` 分支和上游 `master` 分支同步。直接 `git fetch upstream` 即可。

```
$ git checkout master
$ git fetch upstream
$ git merge dev
```

要与远程仓库同步的时候，可以通过 `git push branch_name` 来选择要push到哪个远程仓库上。

参考：[保持fork之后的项目和上游同步](#)

pull request 进行代码review

The screenshot displays a GitHub pull request interface. At the top, a green banner indicates 'Able to merge'. Below this, a green button labeled 'Create pull request' is visible. The interface shows a comparison between two branches, with 2 commits, 2 files changed, 0 commit comments, and 1 contributor. The commits are listed as 'add travis config' and 'add build pass symbol'. The files changed section shows two files: '.travis.yml' and 'README.md'. The '.travis.yml' file shows changes to the language and jdk version. The 'README.md' file shows changes to the build status and environment setup.

当我的远程仓库开发完了一个稳定的特性之后，就可以向主的项目仓库提出一个 `pull request` 的请求了，在里面我们可以清楚地看到代码的改动地方，可以通过这个功能来进行代码review。主仓库的所有者，在review之后，就可以决定要不要合并这个代码，还是继续让提交者进行修改。

部署Travis CI #1

[Edit](#)

[Open](#) Kinpzz wants to merge 2 commits into zhzdeng:master from Kinpzz:master

Conversation 0 Commits 2 Files changed 2 +8 -1



Kinpzz commented just now

Collaborator

+ @

部署Travis CI, 添加.travis.yml 配置文件,完成持续集成。



Kinpzz added some commits 21 minutes ago



add travis config

8c24ffb



add build pass symbol

2c1671a

Add more commits by pushing to the master branch on Kinpzz/movie-booking.



This branch has no conflicts with the base branch

Merging can be performed automatically.

Merge pull request

You can also [open this in GitHub Desktop](#) or view [command line instructions](#).



Write Preview

AA B i



Leave a comment

Attach files by dragging & dropping, [selecting them](#), or pasting from the clipboard.

Styling with Markdown is supported

Close pull request

Comment

Notifications

Unsubscribe

You're receiving notifications because you authored the thread.

1 participant



Lock conversation

Travis CI

CI 全称 Continuous Integration（持续集成），在一个项目中，任何人对代码库的任何改动都会触发CI服务器自动对项目进行构建、测试、甚至部署。可以及时发现问题，随时修复。

而Travis CI是一个在线托管CI的服务，不需要自己搭建CI服务器，可以通过Travis提供的网页服务来配置我们的CI。并且可以直接与GitHub的仓库进行关联，对开源项目免费。

进入Travis CI 官网<https://travis-ci.org>，可以直接使用GitHub账号进行登录。

为一个项目配置好Travis CI的步骤也比较简单，主要就有三步：

1. 激活所需GitHub仓库
2. 将 `.travis.yml` 配置文件加入仓库
3. 使用 `git push` 进行触发

步骤

激活Github仓库

We're only showing your public repositories. You can find your private projects on travis-ci.com.

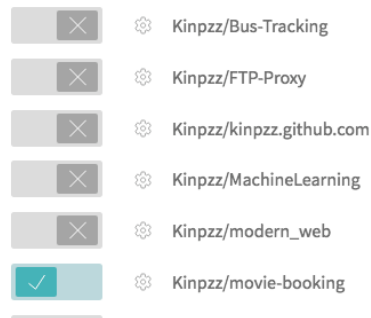
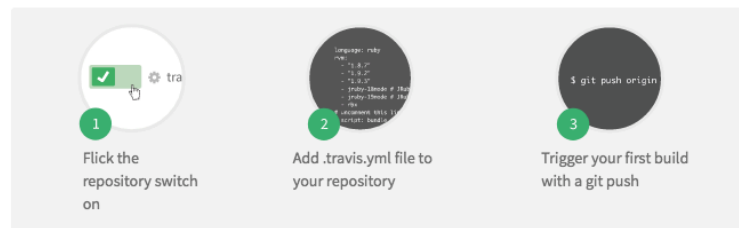
Beta Features

See what's new!

Organizations

You are not currently a member of any organization.

Is an organization missing?
[Review and add your authorized organizations.](#)



交互界面设计的不错，直接点用按钮就完成激活了。

配置.travis.yml文件

`.travis.yml` 文件用来告诉Travis一些项目信息，使用的语言，环境等要求。

```
language: java

jdk:
  - oraclejdk8
```

当我们push到GitHub仓库上Travis一看到java项目，就默认用Maven 3来进行构建，然后执行两个脚本命令

```
$ mvn install -DskipTests=true -Dmaven.javadoc.skip=true
$ mvn test -B
```

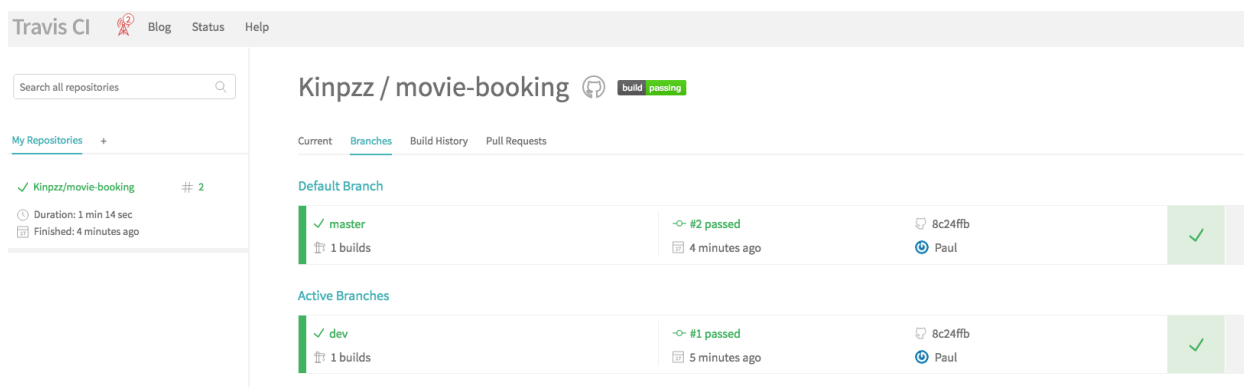
如果我们不使用默认的install和test脚本，可以自己再进行自定义。在 `.travis.yml` 里面进行配置install和test标签。

如在 `.travis.yml` 内添加：

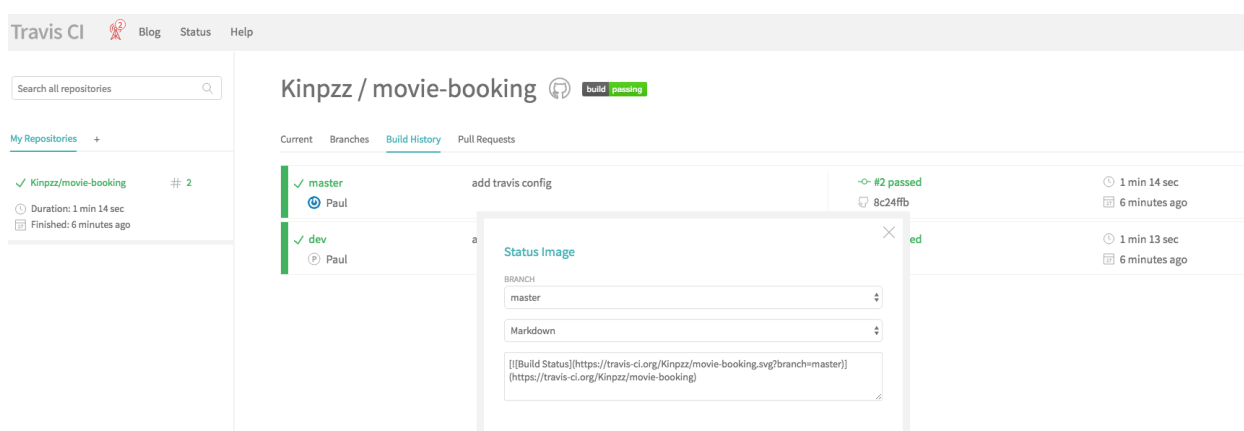
```
install: ./install-dependencies.sh
```

具体可以参考[Travis: Customizing the Build](https://travis-ci.com/docs/travis/getting-started/customizing-the-build/)

之后会通过邮件通知我们构建结果，构建成功在下面可以看见build passing的标志。



还可以通过点击那个标志获取连接，在项目的 `README.md` 文档上显示出一个构建通过的小标志



订你所想 在线电影订票软件

build passing

构建环境

Eclipse + Maven + Spring-boot

运行

方法一：使用maven在命令行

参考

- [使用Travis进行持续集成](#)

- [Pro Git](#)
- [廖雪峰Git教程](#)
- [Travis: Customizing the Build](#)