<http://www.ruanyifeng.com/blog/2017/12/travis_ci_tutorial.html>

持续集成服务 Travis CI 教程

作者： [阮一峰](http://www.ruanyifeng.com/)

日期： [2017年12月19日](http://www.ruanyifeng.com/blog/2017/12/)

[](http://www.zhufengpeixun.cn/main/course/guide.html?ref=ruanyifeng.com)

编写代码只是软件开发的一小部分，更多的时间往往花在构建（build）和测试（test）。

为了提高软件开发的效率，构建和测试的自动化工具层出不穷。[Travis CI](https://travis-ci.org/) 就是这类工具之中，[市场份额](https://github.com/blog/2463-github-welcomes-all-ci-tools" \t "_blank)最大的一个。



本文介绍 Travis CI 的基本用法。用好这个工具不仅可以提高效率，还能使开发流程更可靠和专业化，从而提高软件的价值。而且，它对于开源项目是免费的，不花一分钱，就能帮你做掉很多事情。

一、什么是持续集成？

Travis CI 提供的是持续集成服务（Continuous Integration，简称 CI）。它绑定 Github 上面的项目，只要有新的代码，就会自动抓取。然后，提供一个运行环境，执行测试，完成构建，还能部署到服务器。

持续集成指的是只要代码有变更，就自动运行构建和测试，反馈运行结果。确保符合预期以后，再将新代码"集成"到主干。

持续集成的好处在于，每次代码的小幅变更，就能看到运行结果，从而不断累积小的变更，而不是在开发周期结束时，一下子合并一大块代码。

二、使用准备

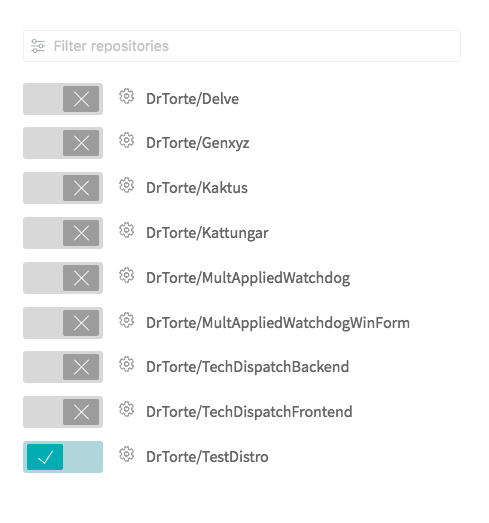
Travis CI 只支持 Github，不支持其他代码托管服务。这意味着，你必须满足以下条件，才能使用 Travis CI。

* 拥有 GitHub 帐号
* 该帐号下面有一个项目
* 该项目里面有可运行的代码
* 该项目还包含构建或测试脚本

如果这些条件都没问题，就可以开始使用 Travis CI 了。

首先，访问官方网站 [travis-ci.org](https://travis-ci.org/)，点击右上角的个人头像，使用 Github 账户登入 Travis CI。

Travis 会列出 Github 上面你的所有仓库，以及你所属于的组织。此时，选择你需要 Travis 帮你构建的仓库，打开仓库旁边的开关。一旦激活了一个仓库，Travis 会监听这个仓库的所有变化。



三、.travis.yml

Travis 要求项目的根目录下面，必须有一个.travis.yml文件。这是配置文件，指定了 Travis 的行为。该文件必须保存在 Github 仓库里面，一旦代码仓库有新的 Commit，Travis 就会去找这个文件，执行里面的命令。

这个文件采用 [YAML](http://www.ruanyifeng.com/blog/2016/07/yaml.html) 格式。下面是一个最简单的 Python 项目的.travis.yml文件。

language: python

script: true

上面代码中，设置了两个字段。language字段指定了默认运行环境，这里设定使用 Python 环境。script字段指定要运行的脚本，script: true表示不执行任何脚本，状态直接设为成功。

Travis 默认提供的运行环境，请参考[官方文档](https://docs.travis-ci.com/user/languages" \t "_blank) 。目前一共支持31种语言，以后还会不断增加。

下面是一个稍微复杂一点的.travis.yml。

language: python

sudo: required

before\_install: sudo pip install foo

script: py.test

上面代码中，设置了四个字段：运行环境是 Python，需要sudo权限，在安装依赖之前需要安装foo模块，然后执行脚本py.test。

四、运行流程

Travis 的运行流程很简单，任何项目都会经过两个阶段。

* install 阶段：安装依赖
* script 阶段：运行脚本

4.1 install 字段

install字段用来指定安装脚本。

install: ./install-dependencies.sh

如果有多个脚本，可以写成下面的形式。

install:

- command1

- command2

上面代码中，如果command1失败了，整个构建就会停下来，不再往下进行。

如果不需要安装，即跳过安装阶段，就直接设为true。

install: true

4.2、script 字段

script字段用来指定构建或测试脚本。

script: bundle exec thor build

如果有多个脚本，可以写成下面的形式。

script:

- command1

- command2

注意，script与install不一样，如果command1失败，command2会继续执行。但是，整个构建阶段的状态是失败。

如果command2只有在command1成功后才能执行，就要写成下面这样。

script: command1 && command2

4.3 实例：Node 项目

Node 项目的环境需要写成下面这样。

language: node\_js

node\_js:

- "8"

上面代码中，node\_js字段用来指定 Node 版本。

Node 项目的install和script阶段都有默认脚本，可以省略。

* install默认值：npm install
* script默认值：npm test

更多设置请看[官方文档](https://docs.travis-ci.com/user/languages/javascript-with-nodejs/" \t "_blank)。

4.4 部署

script阶段结束以后，还可以设置[通知步骤](https://docs.travis-ci.com/user/notifications/" \t "_blank)（notification）和[部署步骤](https://docs.travis-ci.com/user/deployment/" \t "_blank)（deployment），它们不是必须的。

部署的脚本可以在script阶段执行，也可以使用 Travis 为几十种常见服务提供的快捷部署功能。比如，要部署到 [Github Pages](https://docs.travis-ci.com/user/deployment/pages/" \t "_blank)，可以写成下面这样。

deploy:

provider: pages

skip\_cleanup: true

github\_token: $GITHUB\_TOKEN # Set in travis-ci.org dashboard

on:

branch: master

其他部署方式，请看[官方文档](https://docs.travis-ci.com/user/deployment/" \t "_blank)。

4.5 钩子方法

Travis 为上面这些阶段提供了7个钩子。

* before\_install：install 阶段之前执行
* before\_script：script 阶段之前执行
* after\_failure：script 阶段失败时执行
* after\_success：script 阶段成功时执行
* before\_deploy：deploy 步骤之前执行
* after\_deploy：deploy 步骤之后执行
* after\_script：script 阶段之后执行

完整的生命周期，从开始到结束是下面的流程。

1. before\_install
2. install
3. before\_script
4. script
5. aftersuccess or afterfailure
6. [OPTIONAL] before\_deploy
7. [OPTIONAL] deploy
8. [OPTIONAL] after\_deploy
9. after\_script

下面是一个before\_install钩子的例子。

before\_install:

- sudo apt-get -qq update

- sudo apt-get install -y libxml2-dev

上面代码表示before\_install阶段要做两件事，第一件事是要更新依赖，第二件事是安装libxml2-dev。用到的几个参数的含义如下：-qq表示减少中间步骤的输出，-y表示如果需要用户输入，总是输入yes。

4.6 运行状态

最后，Travis 每次运行，可能会返回四种状态。

* passed：运行成功，所有步骤的退出码都是0
* canceled：用户取消执行
* errored：before\_install、install、before\_script有非零退出码，运行会立即停止
* failed ：script有非零状态码 ，会继续运行

五、使用技巧

5.1 环境变量

.travis.yml的env字段可以定义环境变量。

env:

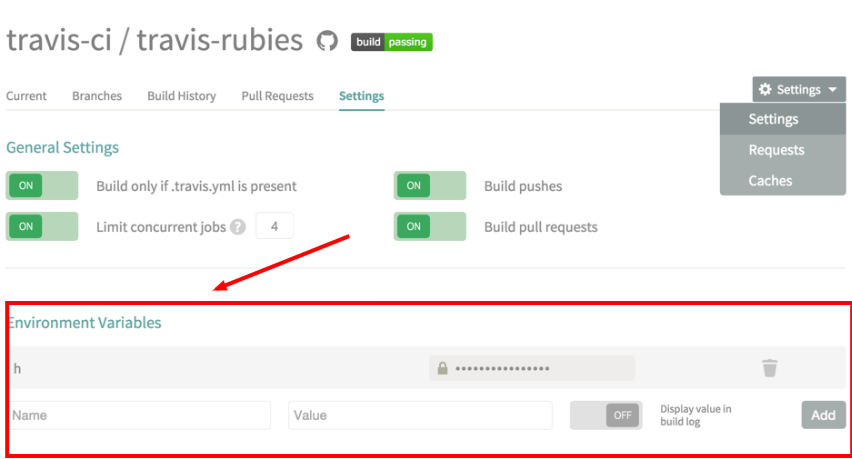
- DB=postgres

- SH=bash

- PACKAGE\_VERSION="1.0.\*"

然后，脚本内部就使用这些变量了。

有些环境变量（比如用户名和密码）不能公开，这时可以通过 Travis 网站，写在每个仓库的设置页里面，Travis 会自动把它们加入环境变量。这样一来，脚本内部依然可以使用这些环境变量，但是只有管理员才能看到变量的值。具体操作请看[官方文档](https://docs.travis-ci.com/user/environment-variables" \t "_blank)。



5.2 加密信息

如果不放心保密信息明文存在 Travis 的网站，可以使用 Travis 提供的加密功能。

首先，安装 Ruby 的包travis。

$ gem install travis

然后，就可以用travis encrypt命令加密信息。

在项目的根目录下，执行下面的命令。

$ travis encrypt SOMEVAR=secretvalue

上面命令中，SOMEVAR是要加密的变量名，secretvalue是要加密的变量值。执行以后，屏幕上会输出如下信息。

secure: ".... encrypted data ...."

现在，就可以把这一行加入.travis.yml。

env:

global:

- secure: ".... encrypted data ...."

然后，脚本里面就可以使用环境变量$SOMEVAR了，Travis 会在运行时自动对它解密。

travis encrypt命令的--add参数会把输出自动写入.travis.yml，省掉了修改env字段的步骤。

$ travis encrypt SOMEVAR=secretvalue --add

详细信息请看[官方文档](https://docs.travis-ci.com/user/encryption-keys/" \t "_blank)。

5.3 加密文件

如果要加密的是文件（比如私钥），Travis 提供了加密文件功能。

安装命令行客户端以后，使用下面的命令登入 Travis CI。

$ travis login

然后，进入项目的根目录，使用travis encrypt-file命令加密那些想要加密的文件。

$ travis encrypt-file bacon.txt

encrypting bacon.txt for rkh/travis-encrypt-file-example

storing result as bacon.txt.enc

storing secure env variables for decryption

Please add the following to your build script (before\_install stage in your .travis.yml, for instance):

openssl aes-256-cbc -K $encrypted\_0a6446eb3ae3\_key -iv $encrypted\_0a6446eb3ae3\_key -in bacon.txt.enc -out bacon.txt -d

Pro Tip: You can add it automatically by running with --add.

Make sure to add bacon.txt.enc to the git repository.

Make sure not to add bacon.txt to the git repository.

Commit all changes to your .travis.yml.

上面的代码对文件bacon.txt进行加密，加密后会生成bacon.txt.enc，该文件需要提交到代码库。此外，该命令还会生成一个环境变量$encrypted\_0a6446eb3ae3\_key，保存密钥，储存在 Travis CI，文件解密时需要这个环境变量。你需要把解密所需的openssl命令，写在.travis.yml的before\_install字段里面。这些都写在上面的命令行提示里面。

--add参数可以自动把环境变量写入.travis.yml。

$ travis encrypt-file bacon.txt --add

encrypting bacon.txt for rkh/travis-encrypt-file-example

storing result as bacon.txt.enc

storing secure env variables for decryption

Make sure to add bacon.txt.enc to the git repository.

Make sure not to add bacon.txt to the git repository.

Commit all changes to your .travis.yml.

详细信息请看[官方文档](https://docs.travis-ci.com/user/encrypting-files/)，实际的例子可以参考下面两篇文章。

* [Auto-deploying built products to gh-pages with Travis](https://gist.github.com/domenic/ec8b0fc8ab45f39403dd)
* [SSH deploys with Travis CI](https://oncletom.io/2016/travis-ssh-deploy/)

（完）