<https://zhuanlan.zhihu.com/p/32656735>

# TarvisCI 全流程使用实践（一）

[史强](https://www.zhihu.com/people/shi-qiang-13)[史强](https://www.zhihu.com/people/shi-qiang-13)

2 个月前

最近开始着迷于github的持续集成。

以前一直把github当成一个 SVN在使用，版本控制和版本管理。后来，发现有些github上有好玩的徽章，于是自己也想弄一个，没想到一下子就不可自拔，github的徽章(Badges)简直就像一个新世界的大门。而打开这个大门的钥匙，就是TarvisCI 。

最近一个月一直在折腾这个东西，乐此不疲，因为真的太强大了。

今天终于利用tarvisCI，打通了，从编译，到代码静态检查，到程序测试，到测试报告回写github的全流程。中间也遇到了不少坑。在这里记录一下。

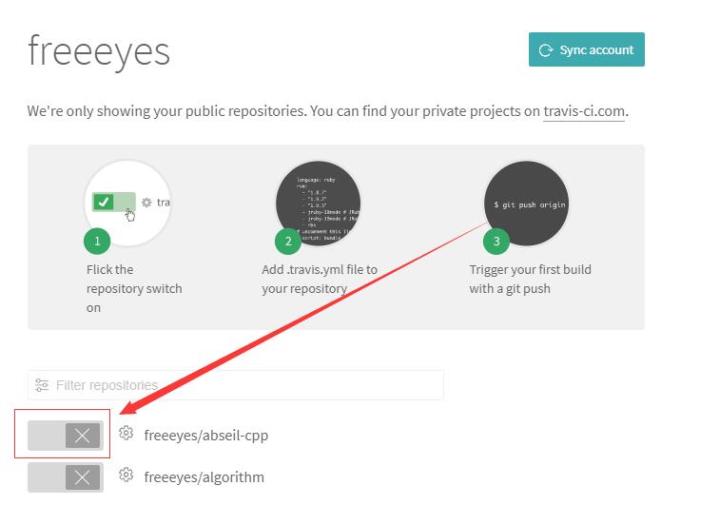
首先先说，编译。

tarvisCI实际提供的是一个暂时的Linux虚拟机（收费版可以有更多的OS选择），你可以把你的代码放在上面去编译执行。当遇到错误的时候，tarvisCI会终止操作，并返回给你一个错误信息。它可以和你的github提交绑定。

首先，你需要登录tarvisCI的网站。（[Test and Deploy Your Code with Confidence](https://link.zhihu.com/?target=https%3A//www.travis-ci.org)）

然后，你需要用你的github账号登录。

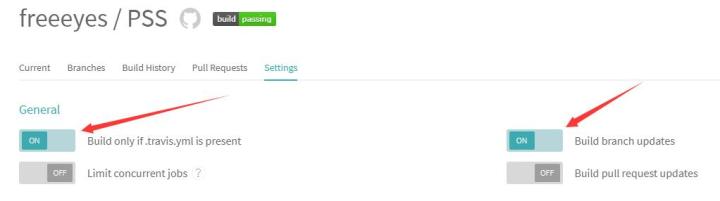
登录成功后，你可以看到你的所有的github工程。



找到你想自动化测试的工程，选择setting



一般，默认的打开和github绑定的选项。一般保持如下两个选项卡打开就行了。



这样，当你每次自动提交的代码的时候，TarvisCI都会帮你自动化编译。当然，这个是依赖脚本的，这个脚本的名称，一般是.travis.yml，也可以是别的名称。但是一定要以.yml后缀为结尾。请注意，这个文件一定要放在github项目的根目录才行。

好了，上述配置完毕。

下面，就讲讲脚本怎么写吧。一个脚本的范例

sudo: required

dist: trusty

language: cpp

os:

- linux

compiler:

- gcc

branches:

only:

- master

env:

global:

- GH\_REF: github.com/freeeyes/PSS.git

- secure: "XXXXXXXX"

before\_install:

before\_script:

- make -j 4

after\_script:

- echo $(pwd)

这是一个标准的脚本。

compiler:  
只的是编译器，因为我的项目是C的，所以用gcc, 如果是java 这里可以指定java的编译器。

branches:  
分支，这里你可以指定你的编译分支，不一定是master，可以是devloper。

env:

这里设置你的全局环境变量。

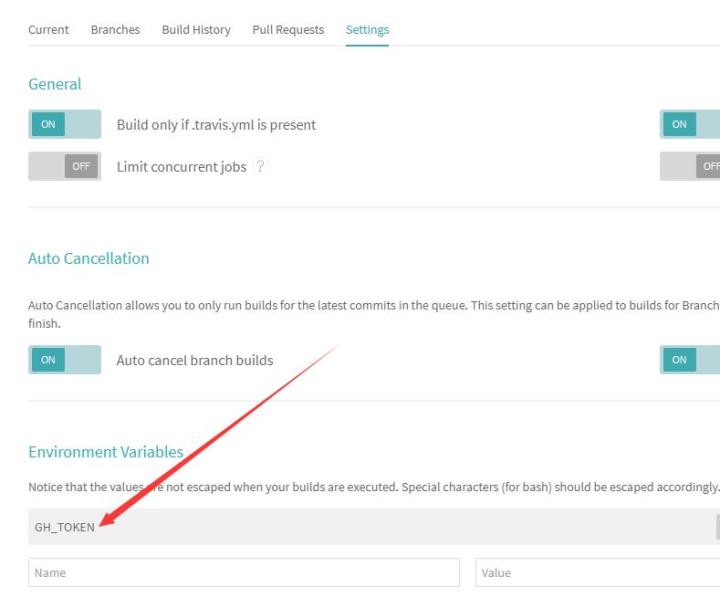
这里要特别强调一个非常有用的东西。GH\_TOKEN。公钥和私钥。

因为有时候，我们需要对某些数据连接进行验证，这里就牵扯到了用户名和密码。比如，我的tarvisCI回调我的github接口写点东西，这时候，我们不可能吧github密码写在脚本里，这样岂不是非常容易被别人盗走？

实际上，tarvisCI已经为你想到了这一点，你可以在tarvisCI的setting界面下，把你的隐私数据设置在这里面，这里可以包括密码，用户名等敏感信息。这些信息，在脚本里，是以全局变量名的形式提供调用，非常的安全。

比如

curl -x -$(GH\_TOKEN)



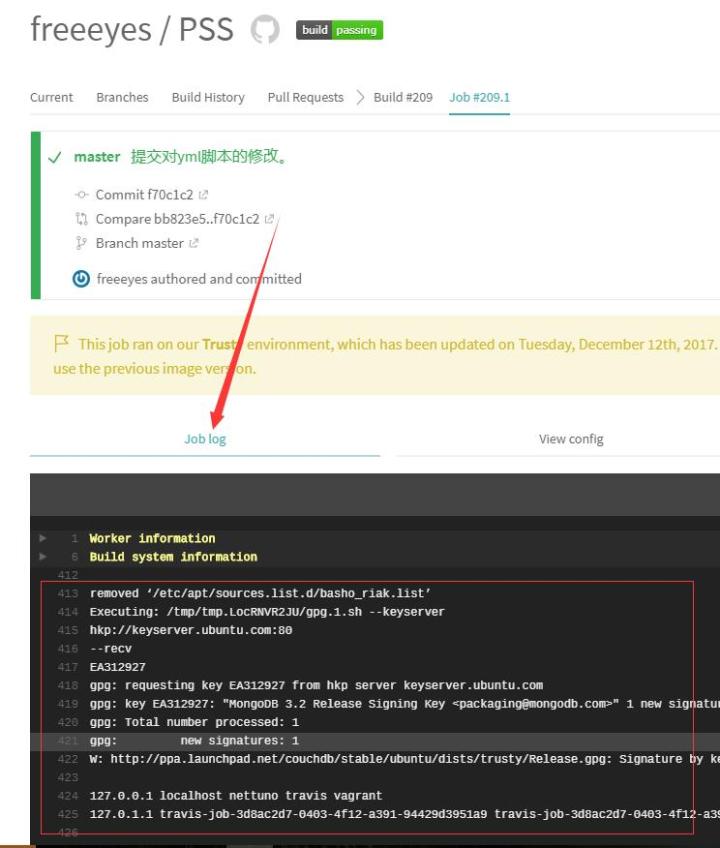
before\_install:

在这里执行你的项目需要在便以前需要的第三方库。或者安装前的准备工作，这里的语法和bash语法完全兼容。

before\_script:  
在这里执行你的工程编译  
  
after\_script:  
在这里，可以添加你的工程编译后的收尾工作，比如make test之类。

写完这个脚本，就可以提交了。

当你提交后，马上就会在tarvisCI下获得执行。



点击箭头的部分，就可以看到结果。

而红框的部分，就是编译日志。在这里你可以看到所有的tarvisCI脚本的执行情况。以及当出现了异常，异常位置和语句是什么。

<https://zhuanlan.zhihu.com/p/32659977>

**TarvisCI 全流程使用实践（二）**

[史强](https://www.zhihu.com/people/shi-qiang-13)[史强](https://www.zhihu.com/people/shi-qiang-13)

2 个月前

tarvisCI既然是一个虚拟机，那么我们就要想办法尽可能的利用它。

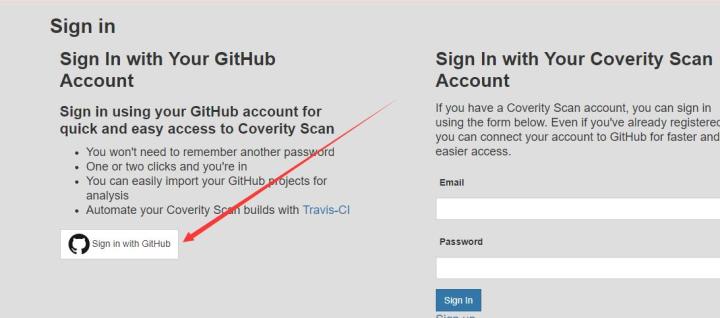
光编译还远远不能满足我的胃口。

那么，来点更棒的东西吧。

我有时候提交代码，经常有笔误的时候，这些小细节有时候不注意，往往累积到一定程度尅安定会爆发出来。于是，我想到了代码静态检查。

[https://scan.coverity.com​scan.coverity.comCoverity Scan - Static AnalysisCoverity Scan - Static Analysis​scan.coverity.com](http://link.zhihu.com/?target=https%3A//scan.coverity.com)

这个网站是和TarvisCI和github无缝集成的。它可以帮你审查你的代码中存在的问题。目前支持所有的主流语言。光C语法就支持规则上千种。

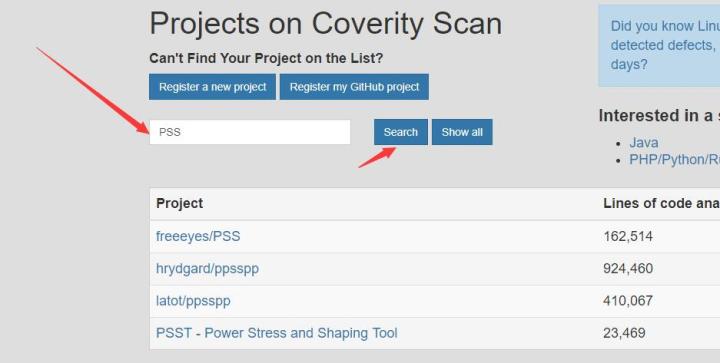


登录网站，在这里使用github账号绑定登录。

这样你可以看到你的github下的所有项目。选择添加工程

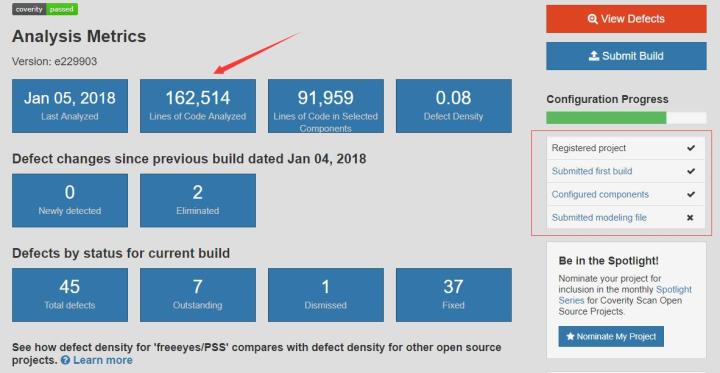


然后输出你的github项目地址后缀就行了。



这样，你的工程就建立了。

在这里，你需要做一些初始化工作。这个工具支持本地编译（下载它的一个巨大的工具包），你可以手动的在你的项目环境下安装测试。当然，这个有点low，既然我们有了免费的tarvis CI，那么就让它去帮我们搞好了。幸运的是，coverity本身就支持tarvisCI。



箭头是你当前项目的代码行数，这里包括了你所有的引用库的代码数量。也就是说，如果你不需要关注你使用的第三方库的代码缺陷，在这里你需要排除它们的路径。这样，代码缺陷检查就不会把扫描的这些缺陷记录在你的项目里面。

红框里的东西，我们必须设定一下，这里有一些要注意。

首先，要注意。如果选择tarvisCI的话，你一定要注意一个地方。



这里有一个secure:

这个一定要记录下来，因为tarvisCI需要这个TOKEN来授权coverity可以接收代码缺陷检查报告。coverity需要根据这个字段来定位，你的这个报告投递给谁。

这个TOKEN是一直有效的，如果你不喜欢，也不介意再生成一个新的。

然后我们来修改一下，tarvisCI的脚本。让它支持代码静态检查。

- os: linux

env:

- secure: "XXXXXXXXXXXXXX(这里是上面页面上显示的TOKEN)"

script:

- export

- echo -n | openssl s\_client -connect scan.coverity.com:443 | sed -ne '/-BEGIN CERTIFICATE-/,/-END CERTIFICATE-/p' | sudo tee -a /etc/ssl/certs/ca-

- if [ "${COVERITY\_SCAN\_BRANCH}" != 1 ]; then

cd $TRAVIS\_BUILD\_DIR/purenessscopeserver/purenessscopeserver/PurenessScopeServer/Linux\_Bin;

$ACE\_ROOT/bin/mwc.pl -type gnuace pss.mwc;

make -j 4;

fi

addons:

coverity\_scan:

project:

name: "freeeyes/PSS"

description: "Build submitted via Travis CI"

notification\_email: XXXXX@163.com;XXXXXX@foxmail.com

build\_command\_prepend: "cd $TRAVIS\_BUILD\_DIR/purenessscopeserver/purenessscopeserver/PurenessScopeServer/Linux\_Bin;$ACE\_ROOT/bin/mwc.pl -type gnuace pss.mwc;make clean"

build\_command: "make -j 4"

branch\_pattern: master

这里添加一个addons:

name: 是你的项目在github上的位置。

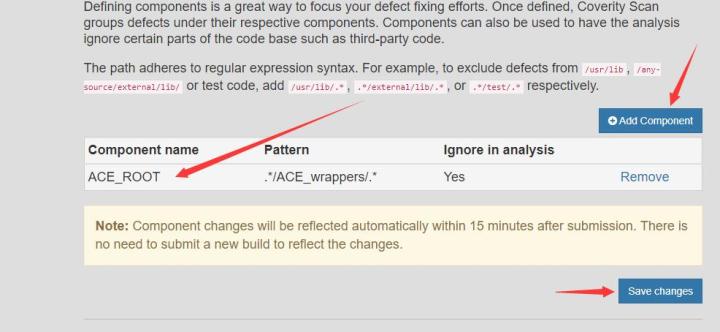
notification\_email: 是当代码缺陷检查完成后，会自动给你发邮件，这里记录你的接收邮箱地址。

branch\_pattern: 这里指定的是你分支。有时候我们不一定要此次都检测，可以在github上设置一个coverity\_scan分支。定期执行，不过我比较懒，喜欢每次提交都给我检测一下，所以直接绑定了master分支。

这里要特别说明一下，coverity免费版是有使用次数的，好像是一周28次，你用完了额度，就得等下周了。当然收费版没有这个限制。

有时候当你频繁提交的时候，会发现没有连续收到报告，当这种情况出现的时候，你去看一下tarvisCI的日志，有时候会有coverity提示。比如你提交的过于频繁啦，下一次执行在XXXX点之后执行等等。免费嘛，拿别人的手短，也就不用过于计较这些了。

最后，如何排除你代码中的第三方库代码呢？



在这个页面上，添加你的忽略路径，英文都很简单的，读一下就会用了。

<https://zhuanlan.zhihu.com/p/32662144>

# TarvisCI 全流程使用实践（三）

[史强](https://www.zhihu.com/people/shi-qiang-13)[史强](https://www.zhihu.com/people/shi-qiang-13)

2 个月前

做到持续集成，光编译，代码静态检查还不够。

我们当然还离不开测试。

其实，如果不是框架级别的服务器，github有很多代码覆盖率检查的标签。因为我写的是服务器，那么，代码覆盖率对我而言并不是特别重要。重要的是，我需要写一个测试客户端，连接我的服务器，发送各种数据包，并检查结果。

这里要说明的是，TarvisCI本身只是一个虚拟机。我们要生成了检测报告，怎么提供出来呢？因为毕竟编译完成后，这个虚拟机就销毁了。我的报告也会随之呜呼哀哉。这个我当然不能接受了，看网上有些人说，tarvisCI关于测试报告没有API接口的讨论，我觉得，既然都做到这一步了，能不能做到更完美呢？没有API也不能阻止我，怎么说来着，"自己动手，丰衣足食。"

那么，怎么动呢？

我的想法初步是，我在TarvisCI上启动我的服务器，然后编译一个测试客户端，去测试这个服务器。将生成的报告，存为一个html。然后把这个html回传到我的github上，这样，每次我自动编译的话，就可以在github上看到我的测试报告结果了。

这里有几个难点。

（1）github本身是需要账号登录的，还是那样，我不可能把我的github密码大摇大摆的写在我的tarvisCI测试脚本里，太危险了。

那么，我需要一种方式，让github通过公钥去授权tarvisCI，然后tarvisCI拿着公钥来提交代码，这样，就算我在脚本里写了公钥的明文，没有私钥一样无法使用，这样彻底解决了明文密码的问题，太COOL了，就这么办。

我找了一台linux服务器。

首先获得一个github的私钥。

curl -X POST -u freeeyes:XXXXXXXXX -d '{"note":"file upload script","scopes":["repo"]}' https://api.github.com/authorizations

{

"id": 154790700,

"url": "https://api.github.com/authorizations/XXXXXXXX",

"app": {

"name": "file upload script",

"url": "https://developer.github.com/v3/oauth\_authorizations/",

"client\_id": "00000000000000000000"

},

"token": "XXXXXXXXXXXXXXXXXXXXX",

"hashed\_token": "XXXXXXXXXX",

"token\_last\_eight": "6364a0b9",

"note": "file upload script",

"note\_url": null,

"created\_at": "2018-01-03T03:12:50Z",

"updated\_at": "2018-01-03T03:12:50Z",

"scopes": [

"repo"

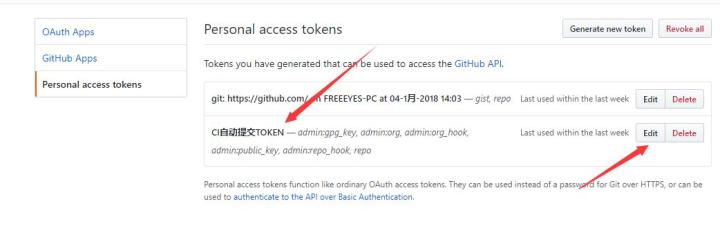
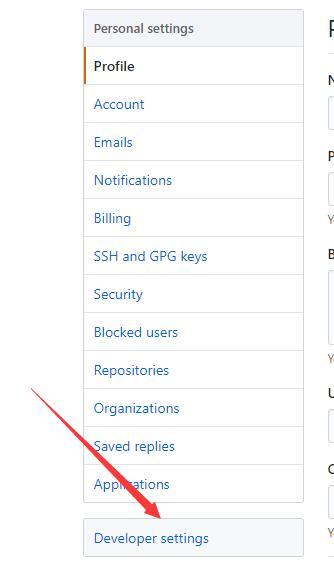
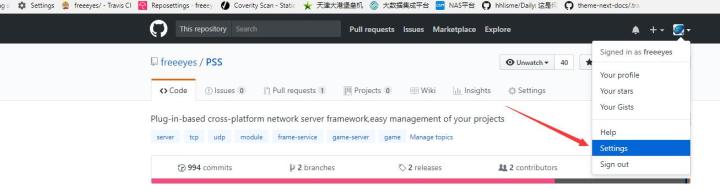
],

"fingerprint": null

}

好了，在这里，拿到了我需要 TOKEN。

然后呢？我们需要给这个TOKEN授权一下。



在github的用户管理设置里，选择你的授权。一定要把写权限加上，因为测试报告，我要提交到github上回来的。没有写权限，tarvisCI执行会有权限问题。

然后找一台Linux，安装一下tarvisCI的客户端，没办法，我们要它的公钥。所以必须借助它的客户端生成。

travis encrypt -r iissnan/theme-next-docs GH\_TOKEN=XXXXXX

Please add the following to your .travis.yml file:

secure: "XXXXXXXXXXXXXXXXXX"

好了，我们得到公钥了。

距离胜利又近了一步。

然后我开始写..travis.yml脚本

sudo: required

dist: trusty

language: cpp

os:

- linux

compiler:

- gcc

branches:

only:

- master

env:

global:

- GH\_REF: github.com/freeeyes/TimeWheel.git

- secure: "XXXXXXX"

before\_install:

before\_script:

- make -j 4

after\_script:

- echo $(pwd)

- echo "wwww" > ./test/aaa.txt

- echo $(ls -l)

- git init

- git config user.name "freeeyes"

- git config user.email "freeeyes@163.com"

- git add -u ./test/aaa.txt

- git commit -m "[ci skip]Update test"

- git push -f "https://${GH\_TOKEN}@${GH\_REF}" HEAD:master

这里剥离一个赶紧的给大家。

这里演示了，我在tarvisCI上模拟生成了一个文件。./test/aaa.txt 并把这个文件上传到github的指定目录下。

这里注意一下。

git commit -m "[ci skip]Update test"

这里一定要加[ci skip]，告诉tarvisCI，不要再试图编译了，否则就是无限死循环。

GH\_TOKEN是什么？请看我在第一章里面讲的tarvisCI隐秘环境变量设置方法。

好了，做到这些，代码报告就会在CI执行测试代码完成后，把我的测试报告上传到我的指定目录，实现完整的测试流程。我只要刷新这个目录，就可以看到我的所有提交测试结果了，

到此，TarvisCI 编译 + 检查 +测试 这个流程全流程走通了。

哦，对了，还有一个自动部署是吧。

这个，我暂时不用，貌似这个还是itarvisCI的pro用户才有的功能。貌似还是收费的，我目前还没有付费的打算，以后再说吧。