

用 Travis-CI 搭建 Github Pages 网站

目录

- [用 Travis-CI](#)
- [用 Travis-CI 部署](#)
- [用 Travis-CI 搭建 Github](#)
- [用 Web 部署静态网站](#)

2015年2月21日

本文档记录了我使用 Travis-CI 部署静态网站的过程。

用 Github Pages 部署静态网站，使用 [Jekyll](#) 生成静态网站。Github Pages 使用 [Jekyll](#) 生成静态网站，使用 `push` 到 Github Pages 仓库，Github Pages 会自动生成静态网站。

用 [Pelican](#) 生成静态网站，使用 `pelican` 生成静态网站，使用 `python` 生成静态网站。

`Windows` `Linux/OSX/Unix-like` `python` `wordpress` `web` `Android` `SL4A` `python` `pelican` `Android` `git`

The diagram illustrates a Travis-CI workflow. It starts with a 'Continuous integration' stage, which includes a 'Build' step (represented by a blue box) and a 'Test' step (represented by a green box). The 'Build' step is connected to the 'Test' step by a blue arrow. The 'Test' step is connected to a 'Deploy' step (represented by a green box) by a green arrow. The 'Deploy' step is connected to a 'Continuous deployment' stage, which includes a 'Deploy' step (represented by a green box) and a 'Rollback' step (represented by a green box). The 'Deploy' step is connected to the 'Rollback' step by a green arrow. The 'Rollback' step is connected back to the 'Build' step by a blue arrow, completing the cycle.

Travis-CI

The diagram illustrates two software development methodologies. On the left, 'Agile Development' is shown as a sequence of four sprints. Each sprint consists of a 'Planning' phase (a box with dots) and a 'Development' phase (a box with horizontal lines). On the right, 'Extreme Programming' is shown as a continuous cycle of 'Planning', 'Development', 'Testing', and 'Deployment'. A 'git commit' label is placed at the end of the development phase in the XP workflow.

Travis-CI ithub repo
github

Travis-CI

1. 在 Github 上创建 repo
 2. 在 Travis-CI 上添加 repo
 3. 在 Travis-CI 上配置 build 脚本



ON

□ Travis-CI □□□□ Github Repo □□□□□

```

    repo: "travis.yml"
    file: ".travis.yml"

```

1 language: python

```
1 language: python
2
3 python:
4   - "2.7"
5
6 before_install:
7   - sudo apt-add-repository ppa:chris-lea/node.js -y
8   - sudo apt-get update
9   - sudo apt-get install nodejs ditaa doxygen parallel
10
11 install:
12   - sudo pip install pelican
13   - sudo pip install jinja2
14   - sudo pip install babel
15   - sudo pip install beautifulsoup4
16   - sudo pip install markdown
17   - sudo npm install -g less
18   - wget "http://downloads.sourceforge.net/project/plantuml
/plantuml.jar?r=&ts=1424308684&use_mirror=jaist" -O plantu
ml.jar
19   - sudo mkdir -p /opt/plantuml
20   - sudo cp plantuml.jar /opt/plantuml
21   - echo "#! /bin/sh" > plantuml
22   - echo 'exec java -jar /opt/plantuml/plantuml.jar "$@"' >>
plantuml
23   - sudo install -m 755 -D plantuml /usr/bin/plantuml
24   - wget https://bintray.com/artifact/download/byvoid/openc
c/openc-1.0.2.tar.gz
25   - tar xf openc-1.0.2.tar.gz
26   - cd openc-1.0.2 && make && sudo make install && cd
..
27   - sudo locale-gen zh_CN.UTF-8
28   - sudo locale-gen zh_HK.UTF-8
29   - sudo locale-gen en_US.UTF-8
30   - sudo locale-gen ja_JP.UTF-8
31
32 script:
33   - git clone --depth 1 https://github.com/farseerfc/pelican-
plugins plugins
34   - git clone --depth 1 https://github.com/farseerfc/pelican-
bootstrap3 theme
```

```
bootstrap theme
```

```
35 - mkdir output
```

```
36 - env SITEURL="farseerfc.me" make publish
```

Travis-CI 環境で Ubuntu 12.04 LTS で python をインストールしようとしたところ、python 2.7 をインストールする際に pip をインストールする before_install ではなく install が必要で build errored となり build fail となった script を修正して build fail を回避した

less.js を ppa を使わずに nodejs でインストールし、less を opencv 1.0.2 で Ubuntu で opencv をインストール(0.4)し、doxygen を opencv をインストールして pelican をインストールして 4 locale をインストールし 4 locale-gen をインストールし Ubuntu を Linux でインストールして Ubuntu をインストールした

.travis.yml を push して github にアップロードして travis に clone して travis に build して passing するまで build した

Travis-CI と Github

travis-ci と Github Pages を使って Github に ssh key をアップロードして github repo を travis にアップロードして key をアップロードした

Github への Personal Access Token

Applications / **New personal access token**

Token description

travis blog push

What's this token for?

Select scopes

Scopes *limit* access for personal tokens. [Read more about OAuth scopes.](#)

☐ repo ⓘ
 ☐ repo:status ⓘ
 ☐ repo_deployment ⓘ

☒ public_repo ⓘ
 ☐ delete_repo ⓘ
 ☐ user ⓘ

☐ user:email ⓘ
 ☐ user:follow ⓘ
 ☐ admin:org ⓘ

☐ write:org ⓘ
 ☐ read:org ⓘ
 ☐ admin:public_key ⓘ

☐ write:public_key ⓘ
 ☐ read:public_key ⓘ
 ☐ admin:repo_hook ⓘ

☐ write:repo_hook ⓘ
 ☐ read:repo_hook ⓘ
 ☐ admin:org_hook ⓘ

☐ gist ⓘ
 ☐ notifications ⓘ

Generate token

? Personal access tokens function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to [authenticate to the API over Basic Authentication](#).

1. Github 上 Personal Access Token App Token を生成する

2. 生成した Personal Access Token Token を `public_repo` 権限を持つ Token として

3. travis へ

2015年2月21日

1. travis encrypt して ruby 上で

2. `repo` `pubkey`

```

1 curl -H "Accept: application/vnd.travis-ci.2+json" https://api.travis-ci.org/repos/<github-id/repo>/key | python2 -m json.tool | grep key | sed 's/.*"key": "(.*)"/\1/' | xargs -0 echo -en | sed 's/ RSA/' > travis.pem
  
```

1 echo -n <github-id/repo> | openssl genrsa -out key.pem 2048
2 openssl req -x509 -newkey rsa:2048 -keyout key.pem -out cert.pem -days 365
3 openssl xargs -0 echo -en "-----BEGIN RSA PRIVATE KEY-----" <key.pem> | openssl rsa -outform -out travis.pem
4 mv travis.pem .travis.yml

5 openssl pubkey -in travis.pem | openssl base64 -w0

```
1  echo -n 'GIT_NAME="Jiachen Yang" GIT_EMAIL=farseerf  
c@gmail.com GH_TOKEN=<Personal Access Token>' | ope  
nssl rsautl -encrypt -pubin -inkey travis.pem | base64 -w0
```

6 echo -n "-----BEGIN RSA PRIVATE KEY-----" | openssl genrsa -out key.pem 2048

7 echo -n "-----BEGIN RSA PRIVATE KEY-----" | openssl genrsa -out key.pem 2048
8 openssl req -x509 -newkey rsa:2048 -keyout key.pem -out cert.pem -days 365
9 openssl xargs -0 echo -en "-----BEGIN RSA PRIVATE KEY-----" <key.pem> | openssl rsa -outform -out travis.pem
10 mv travis.pem .travis.yml

```
1  $ gem install travis
```

11 echo -n "-----BEGIN RSA PRIVATE KEY-----" | openssl genrsa -out key.pem 2048
12 openssl req -x509 -newkey rsa:2048 -keyout key.pem -out cert.pem -days 365
13 openssl xargs -0 echo -en "-----BEGIN RSA PRIVATE KEY-----" <key.pem> | openssl rsa -outform -out travis.pem
14 mv travis.pem .travis.yml

```
1  $ travis encrypt 'GIT_NAME="Jiachen Yang" GIT_EMAIL=farse  
erfc@gmail.com GH_TOKEN=<Personal Access Token>'
```

15 echo -n "-----BEGIN RSA PRIVATE KEY-----" | openssl genrsa -out key.pem 2048
16 openssl req -x509 -newkey rsa:2048 -keyout key.pem -out cert.pem -days 365
17 openssl xargs -0 echo -en "-----BEGIN RSA PRIVATE KEY-----" <key.pem> | openssl rsa -outform -out travis.pem
18 mv travis.pem .travis.yml

```
1  env:  
2    - secure: "long secure base64 string"
```

19 echo -n "-----BEGIN RSA PRIVATE KEY-----" | openssl genrsa -out key.pem 2048
20 openssl req -x509 -newkey rsa:2048 -keyout key.pem -out cert.pem -days 365
21 openssl xargs -0 echo -en "-----BEGIN RSA PRIVATE KEY-----" <key.pem> | openssl rsa -outform -out travis.pem
22 mv travis.pem .travis.yml

```

1  script:
2    - git config --global user.email "$GIT_EMAIL"
3    - git config --global user.name "$GIT_NAME"
4    - git config --global push.default simple
5    - git clone --depth 1 https://github.com/farseerfc/pelican-plugins
6    - git clone --depth 1 https://github.com/farseerfc/pelican-bootstrap3 theme
7    - git clone --depth 1 https://$GH_TOKEN@github.com/farseerfc/farseerfc.github.io output
8    - env SITEURL="farseerfc.me" make publish
9
10  after_success:
11    - cd output
12    - git add -A .
13    - git commit -m "update from travis"
14    - git push --quiet

```

1. 在 `script` 部分添加 `git push --quiet` 命令，确保每次构建成功后都能自动推送到 GitHub。

2. 在 `after_success` 部分添加以下命令，用于构建并推送到 GitHub：

```

- cd output
- git add -A .
- git commit -m "update from travis"
- git push --quiet

```

Web 部署

1. 在 GitHub 仓库中，添加 `Travis-CI` 配置文件 `.travis.yml`，并添加 `build failing` 状态，以及 `Readme.md` 文件。

2. 在 GitHub 仓库中，添加 `Web` 部署配置，并添加 `commit` 事件，确保每次提交都能触发 Travis-CI 构建。

