lcxfs1991 / **blog**

Deploy Using Travis-CI And Github Webhook — webpack doc as an example #19

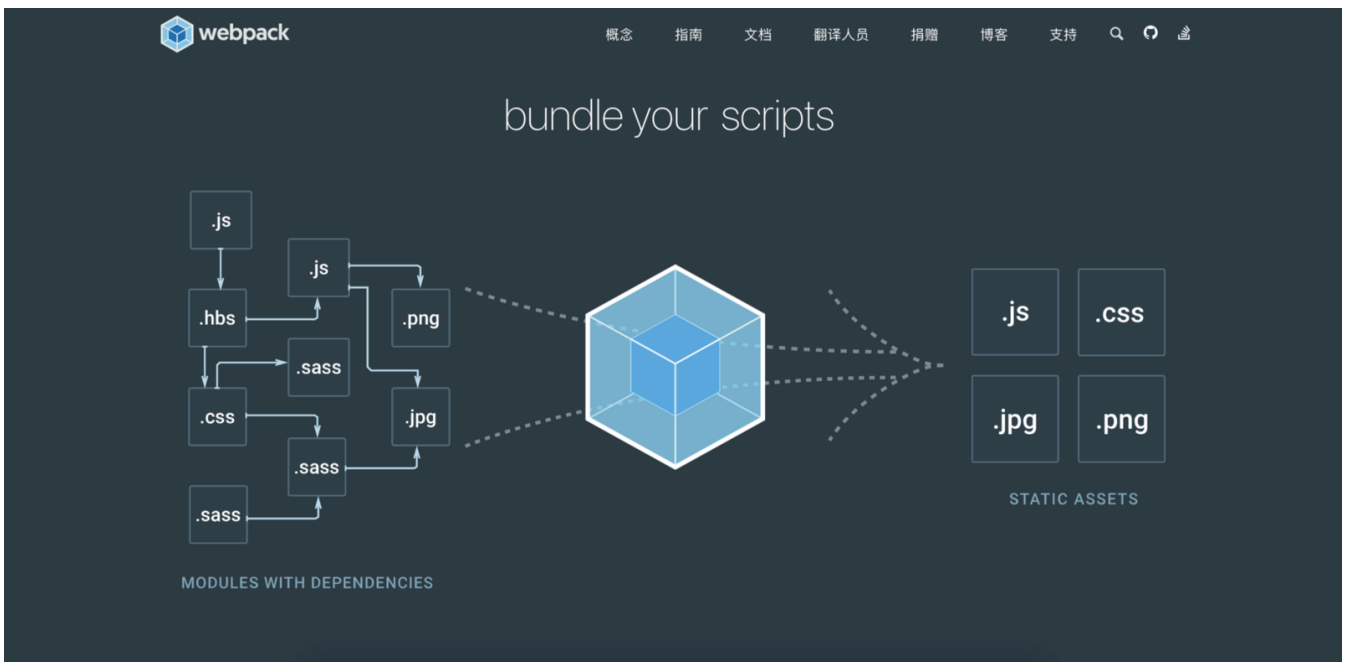
New issue

Open **lcxfs1991** opened this issue on Apr 16, 2017 · 0 comments

lcxfs1991 commented on Apr 16, 2017 • edited ▾ Owner

Overview

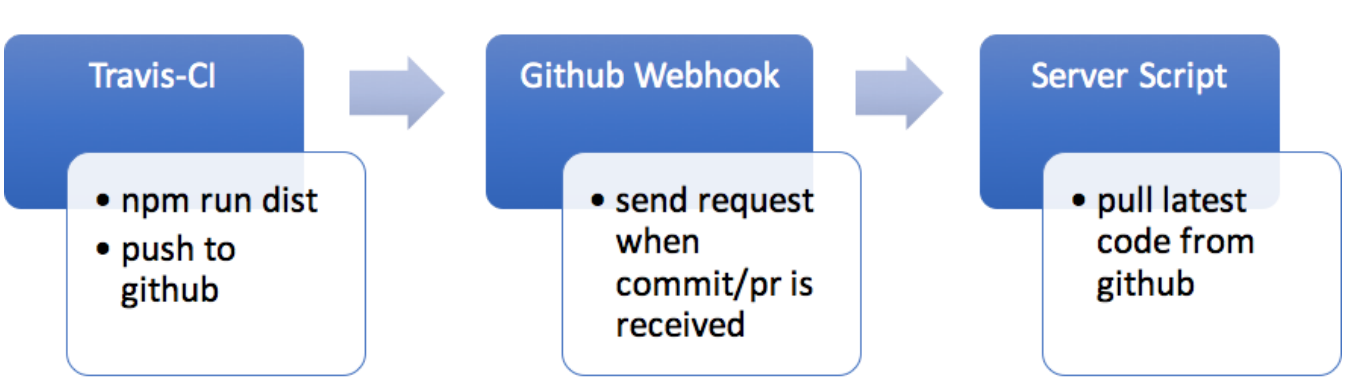
Some friends and I have been running webpack-china for a few months.
After a few months effort, most doc translation job have also been done. We keep tracking the master and you will see Chinese version does not lag behind too much.



> (pretty much the same and http2 is also applied)

However it has been a while that we need to manually deploy the site. With the help of Travis-Ci and Github webhook, we finally make it an auto process.

Travis-CI is used for building your code that you need to publish and push them to your gh-pages branch. Github webhook then takes over the job and sends a request to your hosting server. When your server gets the request, it would run a script to pull the latest code from gh-page. The site then finish all the updating steps.



> (Sample process)

Travis-CI

Here I will use webpack-china/webpack.js.org as an example. The repository is forked from webpack/webpack.js.org. But we have to modify it a bit.

Assignees
No one assigned

Labels
None yet

Projects
None yet

Milestone
No milestone

Notifications

1 participant

```yaml
branches:
  only:
    - cn
language: node_js
node_js:
  - "6"
script:
  - bash ./scripts/deploy.sh
sudo: required
install:
  - npm i yarn -g
  - yarn
  - sudo pip install proselint
```

> (.travis.yml)

This file is the configuration for Travis-CI. Not much difference from other project, branches means which branches will be watched by Travis-Ci. Node.js is used as the major programing language. Node 6.0 environment will be set up. The most important file should be deploy.sh. This file contains the commands used for deployment.

```bash
#!/bin/bash
# see https://gist.github.com/domenic/ec8b0fc8ab45f39403dd
set -e # Exit with nonzero exit code if anything fails

SOURCE_BRANCH="cn"

# Pull requests and commits to other branches shouldn't try to deploy, just build to verify
if [ "$TRAVIS_PULL_REQUEST" != "false" -o "$TRAVIS_BRANCH" != "$SOURCE_BRANCH" ]; then
    echo "Skipping deploy; just doing a build and linting links/prose/js."
    # skip fetching loaders/plugins in cn version
    # npm run fetch
    npm run build
    npm run lint:js
    npm run lint:prose
    npm run lint:links
    exit 0
fi

# Save some useful information
REPO=`git config remote.origin.url`
SSH_REPO=${REPO/https:\/\/github.com\//git@github.com:}

# Fetch loaders/plugins etc. Skip this process in cn version
# npm run fetch

# Run our build
npm run build
```

> (1st part of deploy.sh)

The 1st part of deploy.sh script is used for building the site. Ready releasing code will be put under **build** folder when you run

```
npm run build
```

```bash
# Set some git options
git config --global user.name "Travis CI"
git config --global user.email "ci@travis-ci.org"
git remote set-url origin "${SSH_REPO}"

# Get the deploy key by using Travis's stored variables to decrypt deploy_key.enc
openssl aes-256-cbc -K $encrypted_7562052d3e34_key -iv $encrypted_7562052d3e34_iv -in scripts/de
chmod 600 scripts/deploy_key
eval `ssh-agent -s`
ssh-add scripts/deploy_key
chmod -R 777 node_modules/gh-pages/

# Now that we're all set up, we can deploy
npm run deploy
```

> (2nd part of deploy.sh)

The 2nd part of the script is aiming for pushing code to gh-pages branch. But how does Travis-CI know get the permission to access your repo?
SSH Key does the trick for you. Please follow this guide, Generating a new SSH key to generate SSH Key.
(Please do it under your local repository folder)

example code:

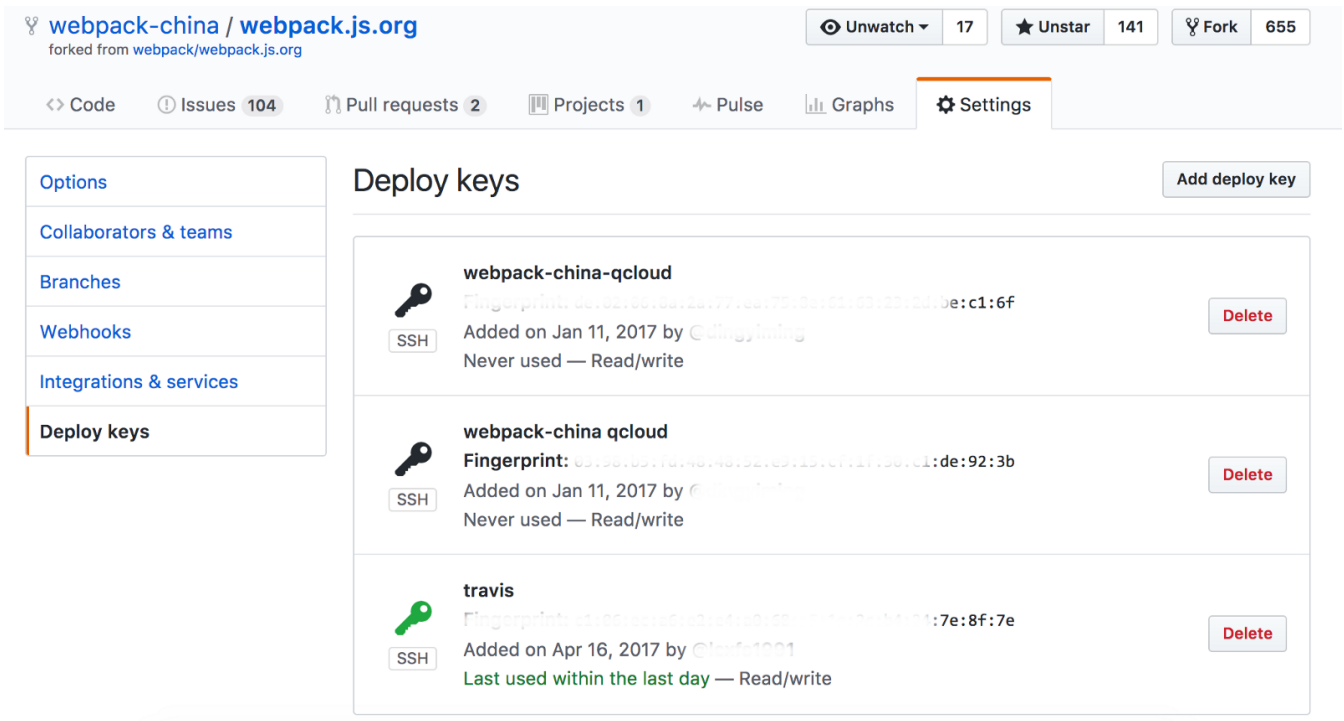
```
ssh-keygen -t rsa -b 4096 -C ci@travis-ci.org

Enter file in which to save the key (/var/root/.ssh/id_rsa): deploy_key
```

When you are asked to enter passphrase, please type enter to skip.

```
Enter passphrase (empty for no passphrase):
```

Then open **deploy_key.pub** and copy whole file content. Add that deploy key to your repository at `https://github.com/<your name>/<your repo>/settings/keys` .



(deploy keys in github)

Next, install travis client tool to upload SSH Key information to the Travis-CI.

After installation, run

```
travis encrypt-file deploy_key
```

```
encrypting deploy_key for webpack-china/webpack.js.org
storing result as deploy_key.enc
storing secure env variables for decryption

Please add the following to your build script (before_install stage in your .travis.yml, f
or instance):

    openssl aes-256-cbc -K $encrypted_7562052d3e34_key -iv $encrypted_7562052d3e34_iv -in
deploy_key.enc -out deploy_key -d

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

Make sure to add deploy_key.enc to the git repository.
Make sure not to add deploy_key to the git repository.
Commit all changes to your .travis.yml.
```

(travis encrypt-file result)

Add the script to deploy.sh under scripts folder and also add deploy_key.enc to scripts folder. Append scripts/ to delopy_key and deploy_key in this script as follows,

```
openssl aes-256-cbc -K $encrypted_7562052d3e34_key -iv $encrypted_7562052d3e34_iv -in scripts/deploy
```

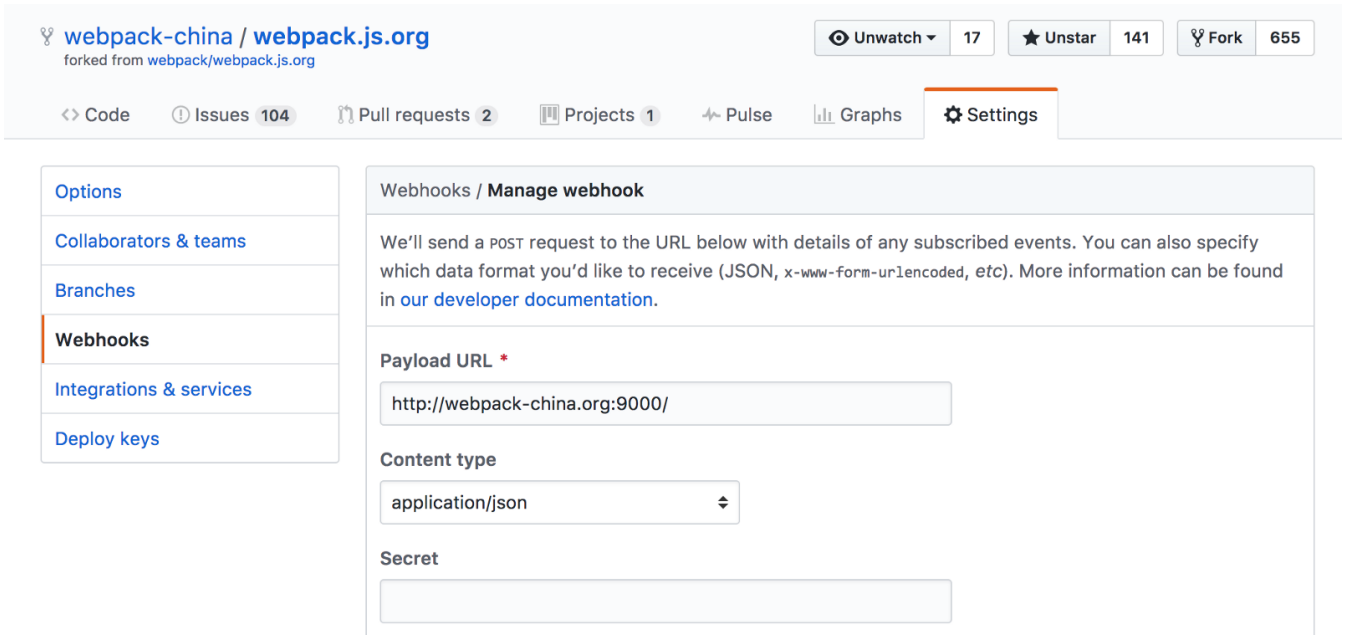
Please do not to upload **deploy_key.pub**.

If it prompts login info, try

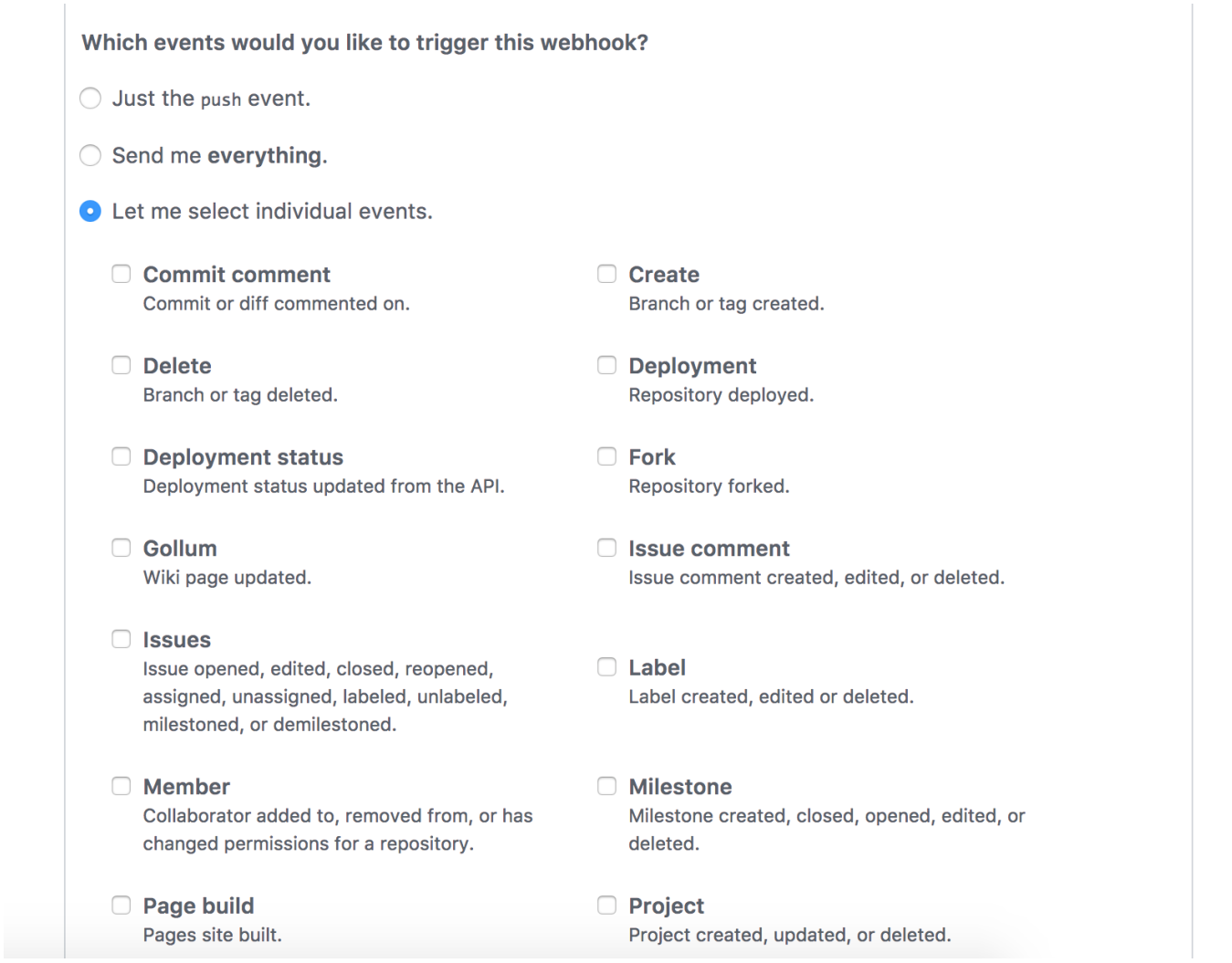
```
travis login
```

Then you can push everything to the repository and Travis-CI will build and push things for you.
One more thing to note is that `npm run deploy` in deploy.sh is used here which use gh-pages library to push code to gh-pages.

GitHub Webhook



(1st part Github Webhook)



(2nd part of Github Webhook)

Add a webhook in your project (1st part of Github Webhook) and you can specify when Github will send the request (2nd part of Github Webhook).

```
const exec = require("child_process").exec;

const http = require("http");
const fs = require("fs-extra");
const path = require("path");

const codePath = '/data/sites/gh-pages';

const deployServer = http.createServer(function(req, res) {
  const updateCommand = `cd ${codePath};sudo git fetch --all;sudo git reset --hard origin/gh-pag

    exec(updateCommand,function (err, out, code) {

    })

    res.writeHead(200, {'Content-Type': 'text/plain'});
    res.write("success");
    res.end();
})

deployServer.listen(9000, () => {
    console.log('listening on 127.0.0.1:9000');
})
```

(app to receive Github Webhook Request)

Then you need to deploy a small app (pm2 is recommended to persist the app process) to respond to Github Webhook.

codePath is the path where accommodate production code from gh-pages branch.

updateCommand,

```
cd ${codePath};sudo git fetch — all;sudo git reset — hard origin/gh-pages;
```

is to fetch all stuff from gh-pages branch and only show the latest record.
Don't forget to configure your **nginx/apache** to serve your static files in codePath.

Reference

Auto-deploying built products to gh-pages with Travis

1

yutingzhao1991 referenced this issue on Apr 23, 2017

文章更新 [**2017-04-16 - 2017-04-22**] #86 Open

lcxfs1991 referenced this issue on Jun 26, 2017

腾讯 **Web** 前端大会**<TFC2017 />** 分享浅析 -- 主会场篇 #21 Open

joshi1983 referenced this issue on Jul 5

Regarding Build System. #599 Open