Continuous Integration for Open Source Projects with Travis-CI
Kyle Tyacke
API Evangelist Akamai
Full Stack Developer
Node Enthusiast
Beer Geek

#OSCON
@GeekGoneNomad
He called her his angel, but she behaved like a devil!

Savage Deception

Lynette Vinet
New Github Project

Travis CI Node Example

A simple example outlining the use of Travis CI for continuous integration with GitHub and Node.js
Pull Requests
Merge Pull Request
Merge Conflict

This branch has conflicts that must be resolved

Use the command line to resolve conflicts before continuing.

You can also open this in GitHub Desktop or view command line instructions.
Stale Pull Requests
What We're Gonna Cover
Continuous Integration (CI)
Travis CI
Preparing for CI
Working with Travis CI
Q&A
Continuous Integration
Continuous Integration

“...a development practice that requires developers to integrate code into a shared repository several times a day. Each check-in is then verified by an automated build, allowing teams to detect problems early.

–https://www.thoughtworks.com/continuous-integration
Continuous Integration

“...a development practice that requires developers to integrate code into a shared repository several times a day. Each check-in is then verified by an automated build, allowing teams to detect problems early."
Continuous Integration

"...a development practice that requires developers to integrate code into a shared repository several times a day. Each check-in is then verified by an automated build, allowing teams to detect problems early."
Continuous Integration

“...a development practice that requires developers to integrate code into a shared repository several times a day. Each check-in is then verified by an automated build, allowing teams to detect problems early.”
Continuous Integration

“...a development practice that requires developers to integrate code into a shared repository several times a day. Each check-in is then verified by an automated build, allowing teams to detect problems early.
Continuous Integration

“...a development practice that requires developers to integrate code into a shared repository several times a day. Each check-in is then verified by an automated build, allowing teams to detect problems early.
Travis CI
Travis CI

...a hosted continuous integration service that's free for open source projects.
Travis CI

...a **hosted** continuous integration **service**
that's free for open source projects.
Travis CI
...a hosted continuous integration service that's free for open source projects.
Travis CI
Open Source
Multi-Language
Pull Requests
Simplified Deployments
Easy
Pull Requests
Pull Request Workflow

1. A pull request is created
2. GitHub tells Travis CI the build is mergeable
3. Hooray! Your build passes!
4. Travis CI updates the PR that it passed
5. You merge in the PR goodness
Pull Request Success

#OSCON

@GeekGoneNomad
Pull Request Success
Pull Request Failure

Add a breaking change to hello_test to show example of Travis failure. #6

ktyacke commented 8 minutes ago

No description provided.

Add a breaking change to hello_test to show example of Travis failure.

Add more commits by pushing to the 3_testing_fail branch on ktyacke/travis-node-example.

All checks have failed
2 failing checks

This branch has no conflicts with the base branch
Merging can be performed automatically.

You can also open this in GitHub Desktop or view command line instructions.
Language Support

Node.js

- "stable"
- "5.1"
- "5"

#OSCON

PHP
Python
Java
Ruby
GO
Objective-C
Perl

Android
C
C#
C++
Clojure
Crystal
Haskell

Julia
Perl6
Rust
Scala
Haxe
Dart
...

@GeekGoneNomad
Adding Travis to Your Project

#OSCON

@GeekGoneNomad
Enable Travis
# THIS PROJECT USES NODE.JS
language: node_js

# BUILD USING THE LATEST STABLE VERSION OF NODE
node Js:
  - "stable"

# RUN OUR TESTS BY CALLING MOCHA
script: mocha
Build Success

#OSCON

@GeekGoneNomad
Build Failure

Pull Request #6 Add a breaking change to hello_test to show example of Travis failure.

- Commit fdb278a
- #6: Add a breaking change to hello_test to show example of Travis failure.

Kyle Tyacke authored and committed

```javascript
> mocha
176  hello
177   \} Returns a string of hello plus the name that was passed.
178
179   \} Returns a string of hello plus the name that was passed:
180     AssertionError: expected 'Hello Dave.' to equal 'Hello Johnny.'
181   at Context.<anonymous> (test/hello-test.js:9:12)

npm ERR! Test failed. See above for more details.
```

The command "npm test" exited with 1.

Done. Your build exited with 1.
0.37s$ git clone --depth=50 --branch=master https://github.com/ktyacke/travis-node-example.git
Cloning into 'ktyacke/travis-node-example'...
remote: Counting objects: 75, done.
remote: Compressing objects: 100% (6/6), done.
remote: Total 75 (delta 1), reused 1 (delta 1), pack-reused 68
Unpacking objects: 100% (75/75), done.
Checking connectivity... done.
$ cd ktyacke/travis-node-example
$ git checkout -qf 55d11fa1be7c186db1bd4b154b3cc343be17534a
This job is running on container-based infrastructure, which does not allow use of 'sudo', so
If you require sudo, add 'sudo: required' to your .travis.yml
See https://docs.travis-ci.com/user/workers/container-based-infrastructure/ for details.
1.60s$ nvm install stable
######################################################################## 100.0%
Checksums empty
Now using node v5.8.0
$ node --version
v5.8.0
$ npm --version
3.7.3
$ nvm --version
0.23.3
install
2.57s$ npm install
npm WARN deprecated graceful-fs@2.0.3: graceful-fs version 3 and before will fail on newer
travis-node-example@1.0.0 /home/travis/build/ktyacke/travis-node-example
├── chai@3.5.0
| └── assert-error@1.0.1

#OSCON @GeekGoneNomad
Travis Build Lifecycle

<table>
<thead>
<tr>
<th>INSTALL</th>
<th>BUILD</th>
<th>DEPLOY</th>
</tr>
</thead>
<tbody>
<tr>
<td>before_install</td>
<td>before_script</td>
<td>before_deploy</td>
</tr>
<tr>
<td>install</td>
<td>script</td>
<td>deploy</td>
</tr>
<tr>
<td></td>
<td>after_success</td>
<td>after_deploy</td>
</tr>
<tr>
<td></td>
<td>after_failure</td>
<td></td>
</tr>
<tr>
<td></td>
<td>after_script</td>
<td></td>
</tr>
</tbody>
</table>
Travis Build Lifecycle

**INSTALL**
- before_install
- install

**BUILD**
- before_script
- script
- after_success
- after_failure
- after_script

**DEPLOY**
- before_deploy
- deploy
- after_deploy
Travis Build Lifecycle

**INSTALL**
- before_install
- install

**BUILD**
- before_script
- script
- after_success
- after_failure
- after_script

**DEPLOY**
- before_deploy
- deploy
- after_deploy
# Travis Build Lifecycle

<table>
<thead>
<tr>
<th>INSTALL</th>
<th>BUILD</th>
<th>DEPLOY</th>
</tr>
</thead>
<tbody>
<tr>
<td>before_install</td>
<td>before_script</td>
<td>before_deploy</td>
</tr>
<tr>
<td>install</td>
<td>script</td>
<td>deploy</td>
</tr>
<tr>
<td></td>
<td>after_success</td>
<td>after_deploy</td>
</tr>
<tr>
<td></td>
<td>after_failure</td>
<td></td>
</tr>
<tr>
<td></td>
<td>after_script</td>
<td></td>
</tr>
</tbody>
</table>
The Travis Client

The Travis Client includes both a command line client and a Ruby library to interface with a Travis CI service. Both work with travis-ci.org, travis-ci.com or any custom Travis CI setup you might have. Check out the installation instructions to get it running in no time.

Table of Contents

- Command Line Client
  - Non-API Commands
    - help - helps you out when in dire need of information
    - version - outputs the client version
  - General API Commands
    - accounts - displays accounts and their subscription status
    - console - Interactive shell
    - endpoint - displays or changes the API endpoint
    - login - authenticates against the API and stores the token
    - monitor - live monitor for what's going on
    - raw - makes an (authenticated) API call and prints out the result
    - report - generates a report useful for filing issues
    - repos - lists repositories the user has certain permissions on
Testing
Testing

language: node_js

node_js:
  - stable

script:
  - mocha
Testing Success
Testing Failure
Linting
Linting

language: node_js

node_js:
  - stable

before_script:
  - npm install -g jscs

script:
  - jscs src
  - mocha
Linting Failure

4 linting broken linting
Commit 29516c0
Compare c341e@e..29516c0
Kyle Tyacke authored and committed

1 code style error found.
The command "jscs src" exited with 2.

$ mocha

hello
Linting Failure
Database Support

- MySQL
- PostgreSQL
- SQLite3
- MongoDB
- CouchDB
- Redis
Databases

services:
  - couchdb

before_script:
  - curl -X PUT http://127.0.0.1:5984/travis-node-demo
  - curl -X GET http://127.0.0.1:5984/_all_dbs
$ sudo service couchdb start
  couchdb start/running, process 2303

$ curl -X PUT http://127.0.0.1:5984/travis-node-demo
  {"ok":true}

$ curl -X GET http://127.0.0.1:5984/_all_dbs
  ["_replicator","_users","travis-node-demo"]
Deployment

- Heroku
- NPM
- PyPi
- Amazon S3
- RubyGems
- Github
Deployment

before_deploy:
  - npm version patch

deploy:
  provider: npm
  email: ktyacke@gmail.com
  api_key:
    secure: XXXXXXXXXXXXX
  on:
    tags: true
    repo: ktyacke/travis-node-example
    all_branches: true
# ENCRYPT AND ADD KEY TO CONFIG
travis encrypt YOUR_API_KEY --add deploy.api_key

# USE CLI TO ADD DEPLOYMENT PROVIDER
travis setup npm
Deployment

The command "jscs src" exited with 0.

Installing deploy dependencies
Preparing deploy
Deploying application

NPM API key format changed recently. If your deployment fails, check your API key in ~/.npmrc.

http://docs.travis-ci.org/user/deployment/npm/

+ travis-node-example@1.0.4
Installing deploy dependencies
Preparing deploy
NPM version: 3.8.3
Authenticated with email ktyacke@gmail.com

Cleaning up git repository with `git stash --all`. If you need build artifacts for deployment, see https://docs.travis-ci.com/user/deployment/#Uploading-Files.

Deploying application
NPM API key format changed recently. If your deployment fails, check your API key in ~/.npmrc
http://docs.travis-ci.com/user/deployment/npm/
~/.npmrc size: 48
+ travis-node-example@1.0.4
Notifications

- Email
- IRC
- HipChat
- Slack
- Webhooks
Notifications

notifications:
  email:
    recipients:
      - foo@bar.com
    on_success: always
    on_failure: always

slack:
  rooms:
    - secure: XXXXXXXXXX
      on_success: always
      on_failure: always
      on_start: always
Notifications

# ENCRYPT AND ADD TOKEN TO CONFIG

travis encrypt "<account>:<token>" --add notifications.slack.rooms
Notifications

#OSCON

@GeekGoneNomad
Notifications

This is the very beginning of the #build-notifications channel, which you created today.
Purpose: Channel to receive build notifications from CI services such as Travis-CI

- Add an app or custom integration
- Invite others to this channel

Today

- ktyacke 2:03 PM
  joined #build-notifications

- ktyacke 2:03 PM
  set the channel purpose: Channel to receive build notifications from CI services such as Travis-CI

- ktyacke 2:03 PM
  added an integration to this channel: Travis CI

- Travis CI BOT 2:06 PM
  Build #52 (73e7497) of ktyacke/travis-node-example@5_notifications by Kyle Tyacke passed in 0 min 37 sec

- Travis CI BOT 2:06 PM
  Build #53 (c1b9471) of ktyacke/travis-node-example@5_notifications by Kyle Tyacke failed in 1 min 2 sec

- Travis CI BOT 2:06 PM
  Build #54 (ed3a83b) of ktyacke/travis-node-example@5_notifications by Kyle Tyacke passed in 0 min 38 sec

- Travis CI BOT 2:06 PM
  Build #55 (944dd35) of ktyacke/travis-node-example@5_notifications by Kyle Tyacke passed in 0 min 45 sec
Build Status Badges

`build` `passing`
#MakeOSFunAgain

#OSCON

@GeekGoneNomad