nodebots

javascript e robótica no mundo real

Wilson Mendes

Nodebots - Javascript e robótica no mundo real

Wilson Mendes

Esse livro está à venda em http://leanpub.com/nodebots-javascript-e-robotica-no-mundo-real

Essa versão foi publicada em 2020-05-14

Conteúdo

Introdução

Esta parte do livro é direcionada a todos que desejam dar os primeiros passos sobre Nodebots ou que tem interesse em aprofundar-se em alguns conceitos que são pouco demonstrados sobre o assunto.

Serão mostrados conteúdos com sensores simples e de baixo custo, porém relacionando os sensores com integrações reais de uma aplicação nodebots, como integração entre API's externas a partir de eventos leitura de alguns dados de sensores, dentre outros.

Além disto o livro aborda tópicos importantes nos quesitos de testes, mostrando boas práticas para a cobertura de testes e como testar com uma boa qualidade e performático, com integrações em serviços externos para automatização de tarefas de validação de qualidade de código e *build*.

Outro ponto abordado é o quesito de evolução da aplicação com foco na arquitetura. Como utilizar padrões de arquitetura de software em qualquer tipo de projeto e como tirar proveito de cada um deles aplicando no seu sistema.

Dando suporte ao seu código em vários sistemas operacionais

Nesta etapa do livro iremos então validar e verificar a cobertura de testes no nosso projeto em diferentes sistemas operacionais, além de habilitarmos diferentes serviços web para melhorias de workflow, como ferramentas de integração contínua e cobertura de código.

Esta etapa é muito importante, pois estas ferramentas nos auxiliam no processo de segurança do nosso código, checando diferentes critérios de aceitação da nossa aplicação de maneira automatizada.

Adicionando servidores de integração contínua ao seu projeto

Como todo projeto de qualidade, o nosso projeto Nodebots vai se preocupar em alguns outros aspectos, como automatização da suíte de testes, build e outras tarefas relevantes ao nosso projeto.

Para isso vamos contar com o auxílio de um servidor de integração contínua. Existem vários no mercado, sendo gratuitos ou pagos, e nesta etapa do livro vamos conhecer um pouco mais do funcionamento e configuração de dois deles: Travis-CI e Appveyor.

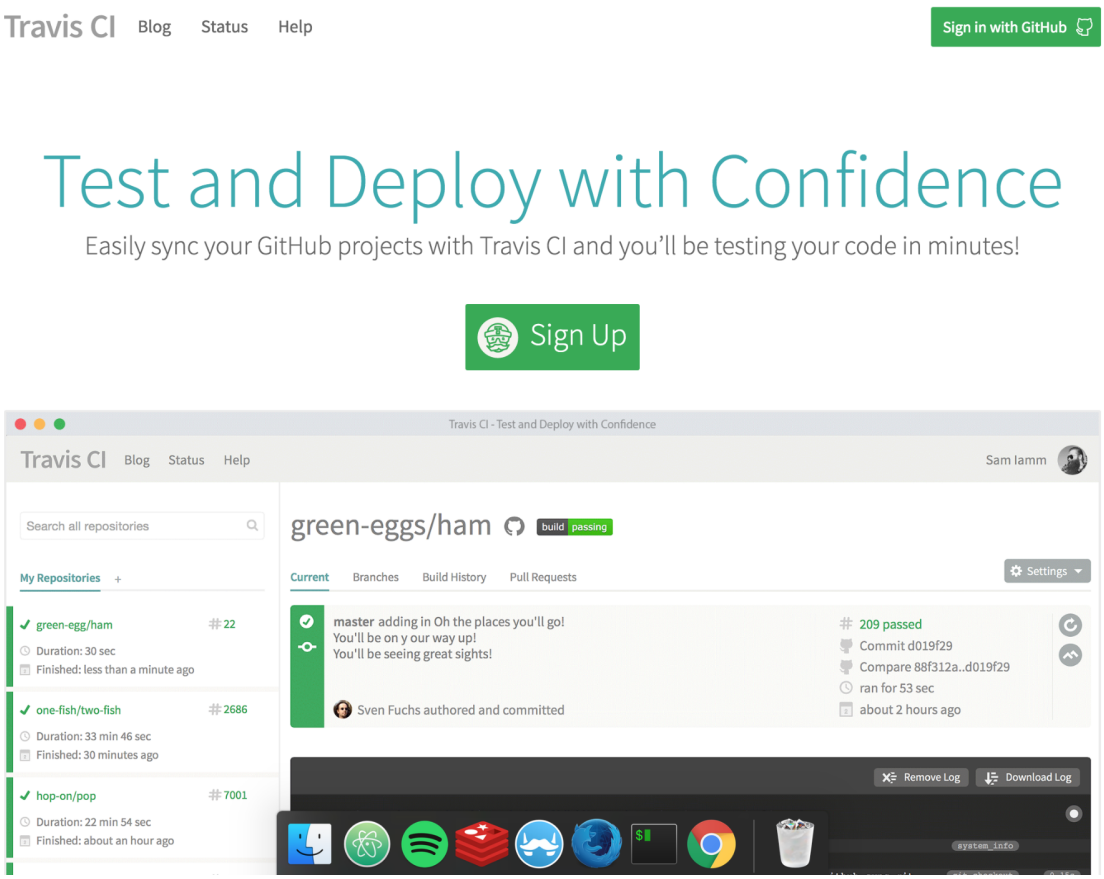
Travis-CI: checando seu código no Linux e OSX

Sabendo que atualmente os sistemas operacionais mais utilizados são Unix/Linux, Windows e OSX vamos criar verificações para cada um deles e para isto entra em cena o Travis-CI.

Ele é um dos mais famosos serviços de integração contínua[1] e auxilia no processo de integração das novas funcionalidades ou correção de bugs do código do atual projeto em vários ambientes, podendo até mesmo efetuar o deploy para produção, caso todos os passos de validação estejam corretos.

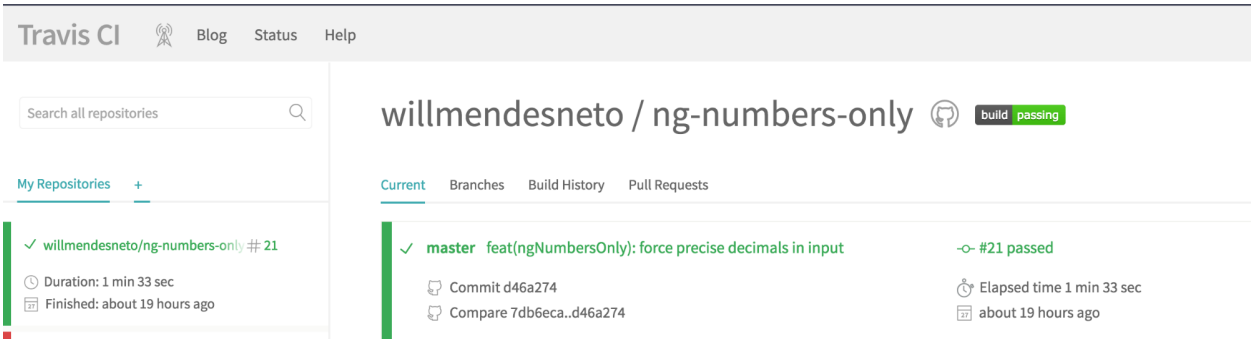
Vamos então ao site oficial do projeto travis-ci[2] e habilitar o acesso utilizando a nossa conta do Github. Clique no botão *"Sign up"* e habilite acesso aos seus repositórios.

[1] http://blog.caelum.com.br/integracao-continua/
[2] https://travis-ci.org/



Site do serviço Travis-CI

Após esta etapa você será redirecionado para uma nova página com todos os seus repositórios. Para adicionarmos um novo basta clicar no ícone "+" ao lado do texto *"My Repositories"*.

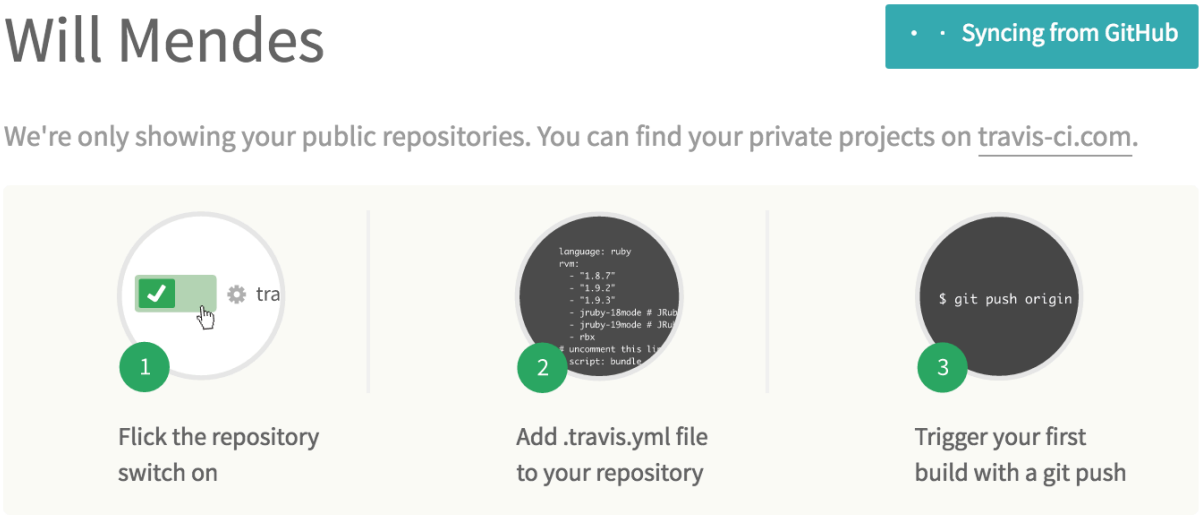


Página de um repositório configurado no Travis-CI

Após esta etapa você será redirecionado para uma nova página com todos os seus repositórios. Para adicionarmos um novo basta clicar no ícone "+" ao lado do texto "My Repositories".

Esta próxima etapa é bem simples já que a página possui um tutorial mostrando cada um dos passos para habilitar a integração do Travis-CI com o seu repositório no Github, como podemos observar

na imagem abaixo.



Sincronizando os repositórios com o serviço

Na mesma página, será listado todos os seus repositórios para que você escolha e habilite a integração do Travis-CI ao seu projeto. Para habilitar basta clicar no botão cinza com um "X" e quando ele mudar a cor para verde quer dizer que esta tudo correu como esperado e o seu repositório está sincronizado com o Travis-CI.

O Travis-CI é totalmente configurável e você pode adicionar informações das mais diversas, sendo elas desde comandos a serem invocados antes, durante ou depois do build, podendo até mesmo configurar os tipos de sistemas operacionais que as tarefas devem acontecer.

Estas configurações ficarão no arquivo `.travis.yml` que ficará na pasta raíz do nosso projeto. Vamos explicar um pouco mais sobre como configurar estas tarefas no Travis-CI.

Ativando o webhook do travis

Primeiramente, no arquivo `.travis.yml`, vamos adicionar o campo `"os"`, com as devidas informações dos sistemas operacionais utilizados para os nossos testes.

```
...
os:
  - linux
  - osx
...
```

Adicionaremos também o campo `"node_js"`, aonde ficarão as nossas informações sobre as versões de NodeJS que as tasks deverão ser utilizados nas nossas tasks. No nosso caso adicionaremos somente uma versão, mas poderíamos adicionar várias outras com base nas nossas necessidades de suporte, por exemplo.

```
...
node_js:
  - '12.16.2'
...
```

O nosso servidor de integração contínua nada mais é do que um container com um sistema operacional completo. Sendo assim podemos também configurar variáveis de ambiente nele. Neste caso adicionaremos a variável `NO_SERIALPORT_INSTALL`, especificando que não devemos instalar o pacote "serialport" neste caso, pois trata-se de um teste que utiliza um `mock` de um board físico.

OBS: A idéia deste livro é de focar nos conceitos relacionados diretamente com Nodebots e integrações com o repositório javascript criado, por isso não explicarei sobre o conceito de `containers`. Caso queira saber mais sobre este conceito utilizado pelo Travis-CI, acesse a página oficial do projeto Docker[3].

```
...
env:
  - NO_SERIALPORT_INSTALL=1
...
```

Podemos também definir o conjunto de tarefas que serão utilizados antes e depois do nosso script do travis. Neste caso utilizaremos `before` para os comandos que devem ocorrer antes do nosso script principal e `after` para os comandos que devem ocorrer depois dos comandos travis, como vocês podem visualizar no trecho de código a seguir:

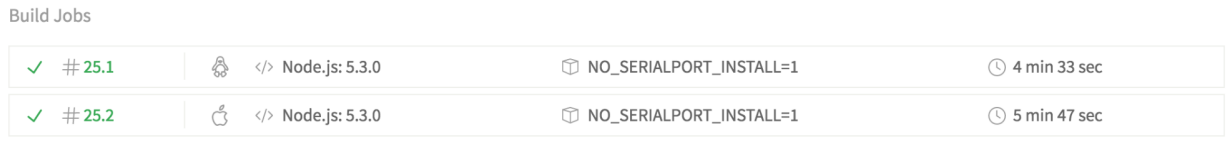
[3] https://www.docker.com

```
...
before_script:
  - 'npm install'


after_script:
  - 'make test'
...
```

Neste caso estamos instalando as nossas dependências e rodando os nossos testes. Tudo isto de uma maneira bem simples e bem definida. O conteúdo do nosso arquivo `.travis.yml` com todas as alterações será o seguinte:

```
language: node_js
os:
  - linux
  - osx
node_js:
  - '12.16.2'
before_script:
  - 'npm install'


after_script:
  - 'make test'
env:
  - NO_SERIALPORT_INSTALL=1
```

Podemos perceber que o build do Travis-CI agora é um pouco diferente, pois estamos rodando o mesmo setup nos sistemas operacionais Linux e OSX, identificados pelos ícones de cada sistema operacional.



Lista de sistemas operacionais utilizados

Com a integração testada, vamos então colocar o badge do travis-ci no nosso arquivo `README.md` do repositório. Com isto você verá uma imagem com o status do build.

```
[![Build Status](https://travis-ci.org/willmendesneto/build-checker.png?branch=maste\
r)](https://travis-ci.org/willmendesneto/build-checker)
```

Com isto terminamos a nossa integração com o servidor de integração contínua Travis-CI e temos toda a nossa suite de testes rodando nos sistemas Linux e OSX. Nesta próxima etapa vamos configurar as mesmas tarefas, mas para serem verificadas no sistema operacional Windows, utilizando outro servidor de integração contínua chamado Appveyor.