Using TravisCI for continuous integration to GitHub organizations

Zachary Bergmann Jun 7, 2017 · 4 min read

Who is Travis? Well…it is more like what is Travis. TravisCI is explained to be "a continuous integration service used to build and test projects hosted at GitHub." TravisCI can be used on repositories that you have hosted on GitHub to perform various tests and operations on your code. Travis can be configured to run linting, perform tests that you have created, and even start up your application to check that everything is healthy and running well. With this information, we as developers can feel more confident that everything being added to our organizations on GitHub will perform correctly and not "break the build."

So, how does TravisCI generally work? Well, I will walk you through the basics of getting Travis set up on a repository and then give a recap, extra info, and places to look for additional information. So, to start with, go to https://travis-ci.org/ and make an account there using your GitHub credentials. Once you have done so, add your desired organizations and local account to TravisCI as using Travis for continuous integration. Now you are able to select a particular repository that you would like to have Travis work on. If you do have any environmental variables that will be needed for testing, etc of your code, they can be added at the TravisCI website for that repository as well using the GUI there. Once you have set all of this up (I assumed that you have a repo already made and on GitHub), go to that repo on your machine.

Now that you are all set at the TravisCI website, you need to add a file to tell Travis exactly what you would like worked on. To do this, create a .travis.yml file at the root of your repo. In this file, I suggest starting with something like this (code below) and then adjusting using the TravisCI documentation which I have found to be very thorough and overall just excellent.

```
language: node_js
node_js:
  - '7'
before_script:
  - psql -c 'create database travis-ci-test;' -U postgres
before_install: if [[ `npm -v` != 4* ]]; then npm i -g npm@4; fi
cache:
  directories:
    -node_modules
sudo: false
```

A bit about the above code: I am using this on a current project that has a PostgreSQL database and is using Node version 7.10.0 / NPM version 4.2.0. From the above file, I think most of this is fairly easy to breeze through until you get to this before_script. That line is just setting up my database (you probably want to replace your database name for "travis-ci-test") with PostgreSQL so that Travis can mimic my database. This step occurs before my project is built so anything that needs to be done before that on your project, add it here. Next, the before_install line is checking the Node and NVM version if "engine" is not specified in your package.json file (I was fine since I specified mine). The before_install section is basically where you can specify additional requirements for a project beyond just the node_modules (normally environment specific, ie Ubuntu). Otherwise this line tells Travis what to have for versions to run my code. Finally, the sudo:false line is just telling Travis to use the latest version of Ubuntu Trusty 14.04 instead of previous versions.

So, to summarize so far, Travis supports many different types of build lifecycle 'hooks'. Here is a list of some of the more commonly used ones that Travis supports:

- before_script Before running the build script
- script How to run the script if you have custom needs
- after_script After running the build script
- before_install Before installing dependencies (used for OS specific packages and extraneous needs)
- install Specify the way to install dependencies (may matter depending on how you bundled your app — Gulp, Grunt, Webpack, etc)
- before_cache Used for cleaning up the cache
- after_success Extra stuff to do after a successful test run by Travis
- after_failure Extra stuff to do after a failed test run by Travis
- before_deploy Steps to run before deploying
- deploy Specify information about how to deploy your code (Heroku, Engine Yard, etc)
- after_deploy Steps to run after the deployment has been completed

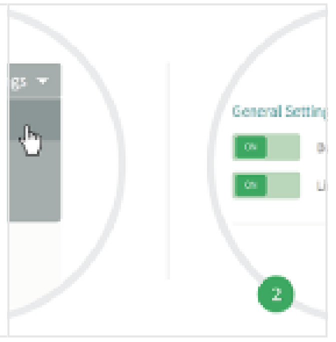
As you can probably tell, TravisCI can basically automate your process from start to end if you would like it to. Travis supports builds with Docker, Cron tasks, all sorts of third party packages, and the list just goes on and on. Just a small amount of time spent on setting Travis up (especially on larger projects with multiple contributors) will assist in keeping a lint-free, operational, and maintainable codebase. With a good linter in used on the development side, Travis just keeps getting better. Add a line such as the ones below to your package.json and Travis will run your tests and check for lints before passing/failing your pull request in GitHub:

```
"test": "mocha run test/test-server.js",
"pretest": "npm run lint",
"lint": "eslint . --ext .json .jsx"
```

Now Travis will run pretest, then test, and more before reporting success or failure back to you. At this point, you can see your pull request in GitHub and determine if it has passed or failed. If it has failed, of course you should go back and fix the code before it should be pulled.

So to summarize, TravisCI is an immensely powerful tool that can make sure that your pull requests get done right and in a manner that keeps your build running. Also, it can be used to automate building, deployment, installs, downloading images, and many other things. I hope you have enjoyed the read and please leave comments below. I look forward to your responses and will continue to share any other interesting information that I come across as I read deeper into and use more of what TravisCI has to offer. Thanks again and here are those links as promised!

Customizing the Build - Travis CI
Travis CI provides a default build environment and a default set of steps for each programming language. You can…
docs.travis-ci.com



Travis CI for Complete Beginners - Travis CI
Note: You can only enable Travis CI builds for repositories you have admin access to.
docs.travis-ci.com

travis-ci/travis-ci
travis-ci - Free continuous integration platform for GitHub projects.
github.com

Continuous Integration · Travis Ci · Nodejs · Github · JavaScript