Previous Next

Unit 3 / Lesson 5 / Assignment 4

Continuous integration using Travis CI

Estimated Time: 1 hour

In this assignment you will use continuous integration (CI) to automatically run your test suite every time you push changes to our GitHub respository. CI is a practice which allows us to automatically build, test, and deploy software every time we make a change to our codebase. This helps to reduce the amount of time you spend on routine development tasks. It also helps ensure that you follow good development practice such as running tests regularly, and deploying code early and often.

You are going to be using a service called Travis CI which allows you to provision a server and run some code every time you push to GitHub. Travis is widely used for open-source projects, and is free to use for public GitHub repositories.

So let's see how it works. First of all make sure that the latest version of your blog code is pushed up to GitHub. Then sign in to Travis from the front page of their website. This will prompt you to give Travis permission to access your GitHub account.

Now that you're signed up you need to activate Travis for your repository. Go to your profile page by clicking on your name in the top-right corner, and then flip the switch next to your blog repository to activate Travis.

With Travis up and running you should create a configuration with settings to connect to Travis' Postgres database. Add the following configuration to *config.py*:

```
class TravisConfig(object):
    SQLALCHEMY_DATABASE_URI = "postgresql://localhost:5432/blogful-tes
    DEBUG = False
    SECRET_KEY = "Not secret"
```

Then you need to tell Travis what to do when you push your code. In the root directory of your blog create a file called *.travis.yml* (notice the initial dot character, indicating that this is a hidden file). Then add the following markup to the file:

```
language: python
python:
    - "3.4"
install: python3 -m pip install -r requirements.txt
env:
    - CONFIG_PATH=blog.config.TravisConfig
before_script:
    - psql -c 'create database "blogful-test";' -U postgres
script:
    - PYTHONPATH=. python3 tests/test_filters.py
    - PYTHONPATH=. python3 tests/test_views_integration.py
    - PYTHONPATH=. python3 tests/test_views_acceptance.py
    # Add any other tests here
```

Here you tell Travis that you are looking to test a Python program. You then specify the versions of Python to test against. Next you instruct Travis to install your dependencies from the same *requirements.txt* that you use for your own installation. You tell Travis to use your new configuration. Then you tell Travis to set up a new Postgres database before the tests are run. Finally you describe the steps required to run each of your tests.

To test it out add and commit your changes to GitHub, then push them up to Travis. If you then go back to the front page of Travis you should see that your repository is now in a queue waiting to be run. If you give it a few seconds (sometimes longer when Travis is busy) then you will be able to see the output of your commands as they are run on the server. When it has finished running your tests you will get an email either telling you that your tests have either passed or failed. If they have failed then try to fix any errors, and push your code back up to GitHub to make Travis run again.

And that's all there is to using Travis and CI. In time you will find that your *.travis.yml* files will become more complex, encompassing more than just running your tests. But that is the joys of continuous integration – pretty much any job which needs to be run regularly on your codebase can be added to your *.travis.yml* file and you will be notified about how it has gone.

What advantages to CI bring to a development process?

> *It allows you to automate any tasks which need running any time the codebase changes. This helps embed good practice such as regular testing, and reduces the time spent carrying out repeated tasks.*

How do you make Travis install your project dependencies?

> *You create a requirements.txt file which contains a list of your dependencies, then in .travis.yml you create an installation rule as follows:* `install: pip install -r requirements.txt`.

How do you tell Travis which commands to run?

> *The commands are supplied in the* `script` *element of the .travis.yml file. This can either be a single command, or a list of commands defined on subsequent lines each starting with a hyphen character.*

☆ ☆ ☆ ☆ ☆ · Report a typo or other issue

✔ Mark as completed

 Previous

Next