Professional Practice Assignment 2 - Deployment Pipeline

Aaron Kimbrell - awk73 - awk73
Jahrell Harris - jah1254 - SpecificAlgorithm
Austin Parker - ajp310 - ajparker96
Austin Markham - ajm712 - ajm712
William Thompson - wtt53 - wtt53

Introduction

The goal of Professional Practice Assignment 2 was to create a continuous deployment pipeline which would help automate the development process from initial coding to final deployment. This was accomplished with a variety of tools used for continuous integration, style linting, and templating. This project was based off of the first practice assignment, to which we added web page versions of each function in place of the original command line implementations. This report will discuss our tools used, along with the benefits and challenges of each, as well as other aspects of our development process.

Deployment Pipeline Tools

Python

We used Python 3.6 as our language for this assignment. Python is a user-friendly, feature-rich language with wide support for many different tools, such as the other tools used in this assignment.

virtualenvwrapper-win

A virtual environment was used to ensure each team member was using the same versions of each of our tools. This is an extremely helpful tool in group development projects since it helps prevent many compatibility issues. Since a virtual environment was already used for the first practice assignment, setting one up for this project was not significantly difficult.

pytest

As with our previous project, we used Pytest for our unit tests as well as a few new end to end tests created for this assignment. Pytest is quite easy to use even in conjunction with other tools such as Flask and provides a reliable framework for testing Python applications.

pylama

For style linting, we used Pylama to test for correct line spacing and indentation in order to make our code style consistent. For instance, we specified line indentations as four spaces. If a different amount of spaces were used, or if tabs were used instead, Pylama would specify that a test failed.

EditorConfig

EditorConfig was used in conjunction with Pylama to help enforce the style practices that pylama expects. For instance, EditorConfig was made to automatically set indented lines to be indented with four spaces instead of tabs or a different number of spaces.

pytest-cov

Pytest-cov was used to generate coverage reports for each file when tests were run. This is useful for making sure that each line of code has been tested in some way.

python-coveralls

We used python-coveralls to compare code coverage against specific commits and generate online HTML coverage reports. These reports are made by simply having coveralls read the .coverage file generated by pytest-cov and making the report based off of that. The created report is in a more visually attractive format and is easier to interpret as a result.

Flask

Flask was used to implement the HTML forms for each function of the application. It is a fairly straightforward tool and can be used to quickly make simple web pages that utilize a Python backend.

Gunicorn

Gunicorn 'Green Unicorn' is a Python WSGI HTTP Server for UNIX. it is forked from Ruby's Unicorn project. We used it to server our app through Google App Engine (GAE).

Jinja2

Jinja2 is a Python templating language we used to bridge our HTML pages with our Python backend. It essentially allowed us to have our HTML pages check for certain conditions within the Python code and return values from the code. This is obviously important for outputting the results of each function to the user.

Cypress

Cyprus is our end-to-end testing tool. This was used to created automated tests that would simulate loading the HTML pages and entering and submitting information. This is quite useful for automatically testing the entire functionality of an application.

Deployment Pipeline Steps

For the deployment pipeline, we used Travis-CI to verify our tests, and after a successful build on origin/master, the build was deployed using Google App Engine's API.
Travis allowed us to verify our builds were correct, since we were developing on Windows and deploying to a Linux server.

GAE provided very easy deployment without having to set up, for instance, Apache or Nginx. The main challenge in setting it all up was the lack of documentation that these services provide. Even when they do provide documentation, it may have problems that are not clearly

documented. For example, Travis' encryption tool for securing GAE's key file will not work on Windows if it is run more than one time (which we did do because we were trying to figure out how the tool works).

Test Cases for Manual Testing

Feature Under Test	Email
Test Case Identifier	PPA2CI 0.1
Purpose of test	This specification describes a test case for testing the "e-mail" feature. This case has a valid email.
Test Input	email: "awk73@msstate.edu"
Expected Test Result	System displays a message with "Email is Validated"
Test Procedure	Step 1: Enter "awk73@msstate.edu" into the email field. Step 2: Press the "Submit" button
Environmental Needs	Chrome Web Browser version 66 or higher
Acutal Test Results	The system displayed the expected results
Remarks	N/A

Feature Under Test	Distance
Test Case Identifier	PPA2CI 0.2
Purpose of test	This specification describes a test case for testing the "distance" feature. This case has valid input.
Test Input	x1: "0" y1: "0" x2: "5" y2: "5"
Expected Test Result	System displays a message with "The shortest distance between your two points is: 7.0711"
Test Procedure	Step 1: Enter "0" into x1 and y1, and enter "5" into x2 and y2. Step 2: Press the "Submit" button
Environmental Needs	Chrome Web Browser version 66 or higher
Acutal Test Results	The system displayed the expected results
Remarks	N/A

Team Roles

Aaron Kimbrell

Team Leader.

Setup and configured the CI and deployment pipeline. Also set up the Flask server and the testing suites that were used on the project.Also wrote the Cypress tests.

Jahrell Harris

My role for the project was to implement the split tip functionality from PPA-1 into to a webpage. I implemented this using Flask web forms, Jinja2, and Pytest.

Austin Parker

Similar to PPA-1, my role was to make sure the BMI functionality was implemented correctly. I designed, implemented, and tested the BMI webpage using Flask web forms, Jinja2 templating, and Pytest for the end-to-end tests.

Austin Markham

My role for the project was to implement the email validation functionality from PPA-1 into to a webpage. I implemented this using Flask web forms, Jinja2, and Pytest.

William Thompson

My role in this project largely involved the retirement functionality which I worked on in the previous practice assignment. I created a Flask web form for the retirement function with Jinja2 Python templating. I also helped write our report.

Google Cloud Platform Usage

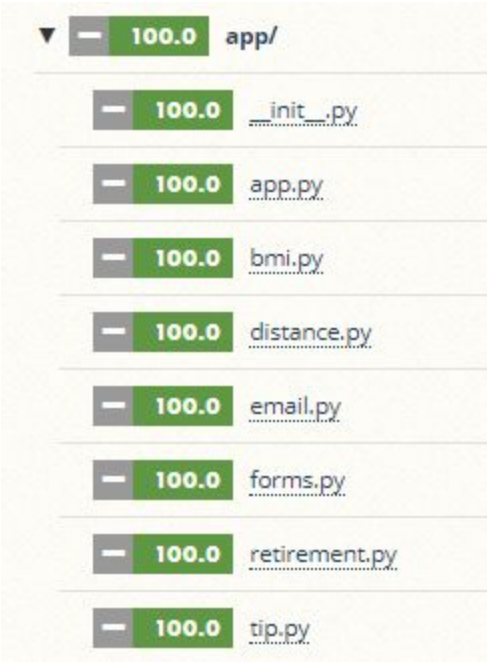
As stated before, one of the challenges with GAE was generating encrypted keyfiles so that Travis could deploy to it. This was solved by using Linux for this step. Another challenge was the lack of documentation for getting GAE to server our app. We did this by using Gunicorn. It is not directly mentioned in GAE's documentation, but it is used in some of their example repos. As explained previously, the biggest benefit of using something like GAE is that we did not have to setup up Apache or Nginx. It also allowed us to continuously deploy to a staging server upon successful builds.

Staging

Production

The difference between staging and production is that Flask's debug suite is enabled on the staging server, but not on the production server.

Code Coverage Report



From: Coveralls