

Introduction

This project focuses its study on the performance of web service frameworks & their development by utilizing a CI/CD methodology & Automated testing tools to determine the most performant and efficient framework between Node.js - Express.js & Python - Flask and also investigates the development efficiency & quality of service gain by utilizing CI/CD methods & techniques.

Aims & Objectives

The aim of the study is to analyse the performance of web service frameworks by using automated testing software & CI/CD methodology to gain insight into efficiency gain from applying this development process & also to determine which framework is the most effective at processing typical REST methods. In order to achieve this, 7 objectives were established which are:

1. Create a CI/CD pipeline to simulate an enterprise level software development team.
2. Create a set of functional & non-functional automated test cases.
3. Develop two rudimentary & identical web services utilizing Node.js with Express.js & Python - Flask.
4. Deploy application to the cloud.
5. Automate testing & iteratively develop the applications.
6. Investigate the integration & performance test results.
7. Analyze the development time saved by utilizing automation.

Research Overview

Poor performing software cost the U.S over 2.84 trillion in 2018 [1].

Facebook, Netflix & GitHub utilise Continuous Integration, Continuous Delivery Continuous Deployment approach and has proven to be effective at reducing errors [2].

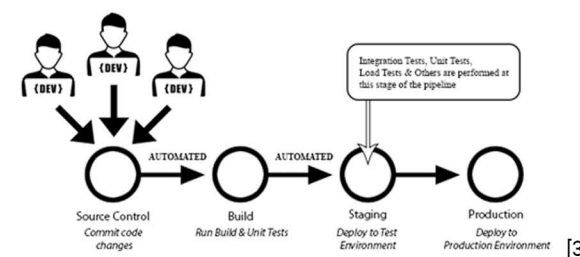


Figure 1: CI/CD architecture diagram

Research Methodology

The research method applied utilised a positivistic philosophy and deductive strategy in order to determine differences between the response time of both frameworks. Results will be analysed using inferential experimental hypothesis testing approach.

The development methodology applied to this study utilised an agile CI/CD approach and Test Driven Methodology for the implementation of the frameworks.

Analysis & Findings

The results of the integration tests show that on average Nodejs completed all aspects of of the fundamental CRUD functions required for the web service.

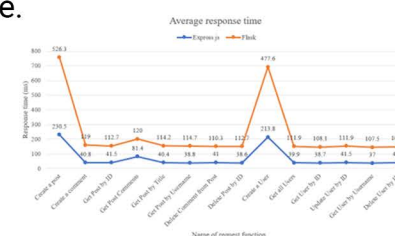


Figure 2: Comparison of average web service response time

Results of the performance test suggest that Express.js could handle a larger user load than Flask

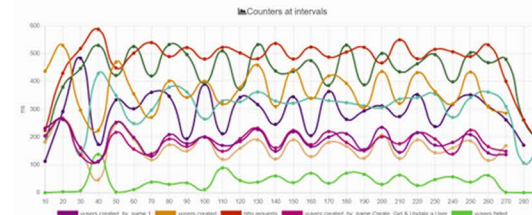


Figure 3: Node.js artillery.io performance statistics

The pipeline saved over 3 hours of manual testing development time, and reduced errors overall as the build would fail if the function did not return the expected output



Figure 4: Jenkins automated testing pipeline

Conclusion

In terms of the performance speed comparison of the frameworks, Node.js completed all requests faster and these results were confirmed by assumed unequal variance t-test at a 0.01 significance level. In this case CI/CD improved the development efficiency, reduced development time and reduced errors occurring.

Reference list

- [1] H. Krasner, 'The Cost of Poor Quality Software in the US: A 2018 Report', p. 44, 2018
- [2] M. Shahin, M. Ali Babar, and L. Zhu, 'Continuous Integration, Delivery and Deployment: A Systematic Review on Approaches, Tools, Challenges and Practices', IEEE Access, vol. 5, pp. 3909–3943, 2017, doi: 10.1109/ACCESS.2017.2685629.
- [3] 'What is Continuous Integration? – Amazon Web Services', Amazon Web Services, Inc. <https://aws.amazon.com/devops/continuous-integration/> (accessed Jan. 04, 2022).