AN ENHANCED BLUE-GREEN DEPLOYMENT FOR REDUCING COST AND APPLICATION DOWNTIME

H.M.D. THILINA JAYAWARDANA

168227V

MSc in Computer Science

Department of Computer Science and Engineering

University of Moratuwa
Sri Lanka

May 2018

DECLARATION

I declare that this is my own work and this MSc Research Report does not incorporate without acknowledgement any material previously submitted for a Degree or Diploma in any other University or institute of higher learning and to the best of my knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgement is made in the text.

Also, I hereby grant to University of Moratuwa the non-exclusive right to reproduce and distribute my thesis, in whole or in part in print, electronic or other medium. I retain the right to use this content in whole or part in future works.

…… ……

H.M.D.T. Jayawardana Date

I certify that the declaration above by the candidate is true to the best of my knowledge and that this project report is acceptable for evaluation for the MSc Research.

…… ……

Dr. Indika Perera Date

ACKNOWLEDGEMENT

I would like to sincerely convey my gratitude to my supervisor Dr. Indika Perera for the guidance and motivation throughout the research. Also my appreciation goes to University LIbrary for providing research materials. I am also thankful for my colleagues and the staff of Pearson Lanka Pvt LTD for their valuable help and support.

ABSTRACT

Application deployment is one of the critical milestones in the software development lifecycle. There are always risks of downtime and failing the new application version. Blue-Green deployment aka A/B deployment is one of the popular web application deployment techniques to mitigate those deployment risks. With the Blue-Green approach, it provides a quick backout plan with an existing set of servers with the previous application version up and running. Even though this has become more popular with the development of cloud infrastructure services, there are some scenarios still this approach brings disadvantages.

In this research, we discuss alternative development approaches in order to address above mentioned concerns while preserving the favorable features which are available in the Blue-Green deployment methodology. It has been considered two alternative approaches for the Blue-Green process without impacting the applications. It has been thoroughly analyzed each alternative approach that we suggest with in order to determine an alternative deployment process for the suitable situation.

Throughout this research, it has been considered Java web application deployment processes as the concerned scenario. As an alternative deployment processes, it has been discussed some of the already existing methodologies and trending novel techniques as well.

It has been proposed two alternative deployment mechanisms comparative to the Blue-Green deployment methodology. The first approach is proposed using the Parallel deployment capability of Apache Tomcat and the second approach is Deployment using Linux containers. Both of these approaches have been tested along with the conventional Blue-Green deployment methodology. The efficiency of each alternative approach has been assessed in a popular cloud environment Amazon Web Service (AWS) considering the practical usage of the solutions.

With this research it has been considered enhancing the existing Blue-Green deployment methodology with the proposed alternative approaches.

By analyzing the results it has been concluded that proposed alternative approaches can be used to enhance the Blue-Green deployment with some pros and cons.

Keywords: Cloud, High-availability, Deployment, Downtime, Release, Web application

TABLE OF CONTENTS

ACKNOWLEDGEMENT ii
LIST OF ABBREVIATIONS x

1. INTRODUCTION 1
1.3. Deployment downtime 4
1.4. The problem and motivation 5
1.5. Outline 7
1.6. Virtualization 7
1.7. High availability architecture 8
1.6. Objectives 8
Reduce deployment downtime 8
Quick rollback and versioning 9
CI/CD support 9

2. LITERATURE REVIEW 10
2.1. Prior work 11
2.1.1. SLA-Aware application deployment 11
2.1.2. A/B Deployment process 12
2.1.3. Canary deployment process 14
2.1.4. Continuous delivery 15
2.1.5. High-available cloud infrastructure 16
2.1.6. Multi-cloud scenarios 16
2.1.7. Database downtime 17
2.2. Literature review summary 18

3. METHODOLOGY 19
3.1. Architecture and scope 20
3.1.1. Versioning 20
3.1.2. CI/CD Integration 22
3.1.3. Deployment process 22
3.1.4. System under test 23
3.2. Evaluation 24

4. SOLUTION ARCHITECTURE AND IMPLEMENTATION 26
4.1. Architecture Introduction 27
4.2. Language and platform 27
4.3. Deployment method 28
4.4. Approach 1 - Parallel deployment with Tomcat server 30
4.4.1. Parallel deployment 30
4.4.2. Proxy interface 30
4.4.3. Clustering 32
4.4.4. Drawbacks 32
4.5. Approach 2 - Using Linux containers 34
4.5.1. Proxy Interface 35
4.5.2. Deployment mechanism 37
4.5.3. NginX 38

5. SYSTEM EVALUATION 39
5.1. Introduction 40
5.2. Load generation 41
5.3. A/B deployment evaluation 41
5.4. Evaluation alternative approaches 42
5.5. Results and discussion 43
5.5.1. A/B deployment deployment with 20 concurrent users 44
5.5.2. A/B deployment deployment with 100 concurrent users 45
5.5.1. Approach 1: using parallel deployment with 20 concurrent users 45
5.5.2. Approach 1: using parallel deployment with 100 concurrent users 47
5.5.2. Approach 2: using Linux containers with 20 concurrent users 49
5.5.2. Approach 2: using Linux containers with 100 concurrent users 51

6. CONCLUSION 54
6.1. Conclusion 55
6.1.1. Alternative approach 1: Parallel deployment using Tomcat 55
6.1.2. Alternative approach 2: Using Linux containers 55
6.1.3. Cumulative conclusions of both alternative approaches 56
6.2. Study limitations 57
6.3. Future works 57

REFERENCES 58
Appendix I - NginX configurations 61
Appendix II - Python helper app source code 63

LIST OF FIGURES

Figure 1.1 Cost of unplanned outage in data centers 3
Figure 1.2 Cost vs duration of unplanned downtime 4
Figure 2.1 A/B deployment 13
Figure 2.2 Canary deployment 14
Figure 2.3 Continuous Delivery Process 15
Figure 3.1 Multiple application versions running in Tomcat 21
Figure 4.1 Approach 1 - Single Tomcat server 31
Figure 4.2 Approach 1 - Tomcat as a cluster 32
Figure 4.3 Approach 2 - Tomcat using Linux containers 34
Figure 4.4 Approach 2 - Proxy interface 36
Figure 4.5 Proxy interface GUI 36
Figure 5.1 A/B deployment Jenkins jobs 42
Figure 5.2 A/B deployment duration 43
Figure 5.3 A/B deployment - Response time for 20 concurrent users 44
Figure 5.4 A/B deployment - Response time for 100 concurrent users 45
Figure 5.5 Approach 1 using parallel deployment - memory usage with 20 users 46
Figure 5.6 Parallel deployment - CPU usage with 20 concurrent users 46
Figure 5.7 Approach 1 using parallel deployment - Response time for 20 users 47
Figure 5.8 Approach 1 using parallel deployment - memory usage with 100 users 48
Figure 5.9 Parallel deployment - CPU usage with 100 concurrent users 48
Figure 5.10 Parallel deployment - Response time for 100 concurrent users 49
Figure 5.11 Parallel deployment duration 49
Figure 5.12 Linux containers - Memory usage with 20 concurrent users 50
Figure 5.13 Approach 2 using Linux containers CPU usage with 20 users 50
Figure 5.14 Linux containers - Response time for 20 concurrent users 51
Figure 5.15 Linux containers Memory usage with 100 concurrent users 52
Figure 5.16 Linux containers CPU usage with 100 concurrent users 52
Figure 5.17 Linux containers - Response time for 100 concurrent users 52
Figure 5.18 Linux containers deployment duration 53

LIST OF TABLES

Table 5.1 A/B deployment - Load summary table for 20 concurrent users 44
Table 5.2 A/B deployment - Load summary table for 100 concurrent users 45
Table 5.3 Parallel deployment - Load summary table for 20 concurrent users 46
Table 5.4 Parallel deployment Load summary with 100 concurrent users 48
Table 5.5 Linux containers - Load summary table for 20 concurrent users 51
Table 5.6 Linux containers - Load summary table for 100 concurrent users 53

LIST OF ABBREVIATIONS

Abbreviation	Description
API	Application Programming Interface
AWS	Amazon Web Services
CD	Continuous Delivery
CI	Continuous Integration
DNS	Domain Name System
DR	Disaster Recovery
HTTP	Hypertext Transfer Protocol
QOS	Quality of Service
SLA	Service Level Agreement
TPS	Transactions Per Second