DevOps Toolchain Required by Efficient Software Development Teams

Ruiyang Ding

DevOps Toolchain Required by Efficient Software Development Teams

Master's Thesis in Computer Science

Parallel and Distributed Systems group
Faculty of Electrical Engineering, Mathematics, and Computer Science
Delft University of Technology

Ruiyang Ding

17th March 2020

Author
Ruiyang Ding

Title
DevOps Tool-chain Required by Efficient Software Development Teams

MSc presentation
TODO GRADUATION DATE

Graduation Committee
TODO GRADUATION COMMITTEE Delft University of Technology

Abstract

In the traditional software development life cycle, development and operation are divided into different departments. The conflict between departments and, besides, the lack of automation usually leads to low software development efficiency and slow software delivery. Thus, the concept of DevOps which combines different departments and automate the process emerges. There are various DevOps platforms provided by different vendors To help the companies migrate to DevOps. To help the company select the suitable DevOps platform, it's important to analyse the needs of the company's side. The thesis project investigates what is the demand from the company on DevOps platform. This is being done by [TODO: How to get this, literature survey?]. Besides, the project compares main DevOps platforms which are being used in the projects of Eficode. The first part of the comparison is from the quantitative perspective. We compare the performance and quality of the software delivery pipeline of tested platforms by deploying a sample [TODO: What kind of application] application through the same continuous delivery model deployed on the different DevOps platforms. Furthermore, we also compared the cost of the different platform in the same testing. The second part of the comparison is from the functionality perspective. In this part, whether these platforms satisfy the need from the team which adopting DevOps is being analysed.

Preface

TODO MOTIVATION FOR RESEARCH TOPIC

TODO ACKNOWLEDGEMENTS

Ruiyang Ding

Delft, The Netherlands
17th March 2020

Contents

Chapter 1

Introduction

1.1 Problem Statement

DevOps is a concept which emerged in recent years. The term ”DevOps” is created by Patrick Debois in 2009, after he saw the presentation ”10 deployments per day” by John Allspaw and Paul Hammond.[8] DevOps is the combination of practices, and culture which aims to combine separate departments(software development, quality assurance and the operation and others) in the same team, in order fasten the software delivery without risking high software quality.[1][6]

For a software team which will employ DevOps, the team must first understand what are the DevOps practices needed. This problem leads to the RQ1 of this paper. Usually, the practices that the team need to adopt help to employ DevOps includes: automated testing and deployment, monitoring, team-working and continues integration etc.[7][9].

DevOps strongly rely on tools. There are specialised tools exist for helping teams adopt different DevOps practices[9]. There are different categories of tools used for different parts of the DevOps practice. For example, Jenkins for continuous integration, Ansible for process automation, sonarqube for code analysis, Slack for team communications etc. It could be interesting if we could investigate what kind of tools does it exist, and how those tools could help teams in DevOps different practice. The research could include how these tools could combine as well. This leads to RQ2.

In software engineering, the toolchain is a set of tools which combined for performing a specific objective. Thus DevOps toolchain is the integration between tools that specialised in different aspect of the ecosystem, which support and coordinate the DevOps practices. The DevOps toolchain could assistant business in creating and maintain an efficient software delivery pipeline, simplify the task and further achieve DevOps.[3][4]

The deployment of toolchain could be on-premise, which means the software development team does the deployment and integration between the tools by themselves. This could: first, provides the maximum freedom for the team to choose the

tools, and secondly, allow team customise the toolchain according to their needs. However, the downside is that the need for extra time for deployment and maintenance. Besides, the cost is hard to calculate and control.

With the development of cloud technologies, now some vendors starting provides the DevOps toolchain under a single application with the concept of Software as a Service(SaaS). A good example is GitLab CI [1]. GitLab CI provides a complete set of tools which covers the whole lifecycle. The toolchain is delivered as a single platform that allows development teams to start using DevOps toolchain without the pain of having to choose, integrate, learn, and maintain a multitude of tools. [2]

So the RQ3 of this thesis project focuses on the comparison between these 2 kinds of toolchains. We will build a DevOps toolchain with the popular tools used in the industry, the deployment and integration of the toolchain will be on-promise. We will also try to justify why do we select a certain tool for each section. This self-built toolchain will be used to compare with commercial singe application DevOps toolchain. We will pick the most popular singe application DevOps toolchain for comparison. In the comparison, we will simulate the same DevOps lifecycle of a demo Spring Boot web app in these 2 toolchains. The perspective of comparison between these toolchains will include:

- Development time: The time spend for implements the toolchain and set up the whole DevOps pipeline.
- Cost structure: The total cost for using the toolchain. For self-built toolchain, it will also include the cost decomposition (for different tools).
- Flexibility: How much freedom can you add/change tools in the toolchain.
- Scalability: How easy to scale the toolchain for the larger project.
- Performance: The performance of the Continues Delivery pipeline, for the same task, how long will it take for the whole process?

From this part of the study, we could make a full comparison between on-premise toolchain and the single application toolchain. For software development teams, it could provide better insights into how to select the DevOps toolchains.

1.2 Research Objectives and Questions

- RQ1: What kind of practices a software development team need to employ DevOps.
- RQ2: What kind of DevOps tools could be included in the toolchain that would help the team implements the practices mentioned in RQ1.
- RQ3: How does the single platform toolchain compared with the on-premise toolchain we build?

[1] https://about.gitlab.com/stages-DevOps-lifecycle/

1.3 Thesis Structure and Main Contributions

In Chapter 2, we will introduce concepts within the scope of DevOps. We will also include the concepts in cloud computing which is related to our research. Chapter 3 is focuses on the literature analysis of DevOps practices. Chapter 4 is focuses on the tools which helping apply the DevOps practices. Chapter 5 focuses on the implementation of DevOps toolchains and the experimentation for comparison between 2 kinds of toolchains. We will summarize our research on the Chapter 6.

The main contributions of this paper are:

- We conduct a literature research on the practices that are necessary for a team to employ DevOps(Chapter 3). We provide a study on the DevOps tools and their mapping to different DevOps practices. This part of research could help the software team which is going to employ DevOps understand the practices needed. Besides, the research gives them a clearer scope on the tools needed for implementing the practices(Chapter 4).

- We give the overview on 2 different types of DevOps toolchain. We also implements demo prototypes for each type of the toolchain, and conduct experiments with these prototypes. The experiment result shows the comparison between different toolchains. It could help team understand which toolchain cloud be selected according to the needs(Chapter 5).

Chapter 2

Background and Concepts

In this chapter, we will introduces several main concepts that related to our study.

2.1 Agile software development

According to the Manifesto for Agile Software Development, compared with traditional software development, the agile software development value these aspects: [5]

- Individuals and interactions over processes and tools.
- Working software over comprehensive documentation.
- Customer collaboration over contract negotiation.
- Responding to change over following a plan.

2.2 Continuous Integration and Delivery

2.3 DevOps

2.4 Tool as Service

Chapter 3

Conclusions and Future Work

3.1 Conclusions

TODO CONCLUSIONS

3.2 Future Work

TODO FUTURE WORK

Bibliography

[1] Devops - wikipedia. https://en.wikipedia.org/wiki/DevOps. (Accessed on 02/24/2020).

[2] The devops lifecycle with gitlab — gitlab. https://about.gitlab.com/stages-devops-lifecycle/. (Accessed on 03/11/2020).

[3] Devops toolchain - wikipedia. https://en.wikipedia.org/wiki/DevOps_toolchain. (Accessed on 03/11/2020).

[4] Toolchain - wikipedia. https://en.wikipedia.org/wiki/Toolchain. (Accessed on 03/11/2020).

[5] Kent Beck, Mike Beedle, Arie Van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, et al. Manifesto for agile software development. 2001.

[6] Christof Ebert, Gorka Gallardo, Josune Hernantes, and Nicolas Serrano. Devops. *Ieee Software*, 33(3):94–100, 2016.

[7] Ramtin Jabbari, Nauman bin Ali, Kai Petersen, and Binish Tanveer. What is devops? a systematic mapping study on definitions and practices. In *Proceedings of the Scientific Workshop Proceedings of XP2016*, pages 1–11, 2016.

[8] Gene Kim, Jez Humble, Patrick Debois, and John Willis. *The DevOps Handbook:: How to Create World-Class Agility, Reliability, and Security in Technology Organizations*. IT Revolution, 2016.

[9] Liming Zhu, Len Bass, and George Champlin-Scharff. Devops and its practices. *IEEE Software*, 33(3):32–34, 2016.