Enhancing Software Deployment Release Time Using DevOps Pipelines

Zeinab Shoieb, Laila Abdel-Hamid, Manal A. Abdel-Fattah

Abstract— DevOps is a software development method that focuses on communication, integration, and collaboration among IT professionals to enables the rapid deployment of products. DevOps is a culture that promotes collaboration between Development and Operations teams. This allows deploying code to production faster and in an automated way. It helps to increases an organization's speed to deliver applications and services. It can be defined as an alignment of development and IT operation. In previous work, we have found that there is a high degree of divergence in how continuous integration, specifically, is interpreted and implemented. This confusion is not limited to continuous integration but is arguably even more pronounced in the case of continuous delivery (CDE) and continuous deployment terms which are often used interchangeably. Although challenges and solutions for CDE adoption have been discussed in the literature, little work is particularly focused on implementing CDE. Furthermore, existing reviews could not find highly relevant scientific literature on CDE implementation.

Index Terms— Continuous Delivery, Continuous Deployment, Continuous Integration, Continuous Release, DevOps and Pipelines.

1 Introduction

Market needs are changing continuously and always required providing products faster to market due to competition among software companies which puts an increasing pressure to deliver new features extremely fast. Eliminating waste is the first principle that explains the Waste as any unnecessary activities that add cost or time without adding value to the customers (partially completed work, extra Features, extra processes, task switching, Handoffs, delays and defects). DevOps appeared as a result of integration of development and operators team members to close the gap between developers and operators [2] and to increase speed of new software releases and reduce time to respond to customer needs and changes. These teams have a different goal, development teams are interested for deliver new features and operations teams are interested for stability [17].

Continuous Integration (CI) is a software development practice where members of a team integrate their work frequently, usually each person integrates at least daily - leading to multiple integrations per day. Each integration is verified by an automated build (including test) to detect integration errors as quickly as possible [14]. It aims to continuously integrate source code to the main branch [10], the developer to commit the code several times in a day followed by automatic build and test and immediate feedback to the developer whenever any bug is encountered. If no bug is encountered, the committed code is pushed to the production [3]. Continuous Integration is essential to be able to establish a Continuous Delivery Continuous Deployment (CD) it means whenever a feature is ready for delivery, it can be released immediately – given that a suitable development and deployment infrastructure is in place [2]. It is an operations practice where release candidates evaluated in continuous delivery are frequently and rapidly placed in a production environment, the nature of which may differ depending on technological context. This often, but not necessarily, implies making it generally available to users, while in other contexts it is not even applicable as a concept [1].The Continuous Deployment pipeline is the set of tools, which enables the workflow to deliver source code from the version control system through the build system and the tests to production in a continuous matter. The code shall be ready for being deployed to target after passing through the Continuous Deployment pipeline, the first step in the Continuous Deployment pipeline is Continuous Integration. The purpose of Continuous Deployment is to achieve the ability to be able to deploy code to the customers as soon as new code has been produced [10]. Continuous deployment typically deploys the new build into production after successful continuous integration pass, but the deployment can also consist of automated tasks as deploying virtual machines, and installing and configuring operating system, supporting software, and libraries. Continuous deployment can automatically scale up and down the deployment on cloud computing platforms depending on the demand [2]. As 29% believed that continuous deployment would help with infrequent releases [18].

Continuous integration and continuous deployment are considered key parts of DevOps. Continuous integration process typically includes automated build, unit tests, and integration tests for modified source code that is automatically pulled from a revision control system [2].

Continuous Delivery (CDE) is a software engineering approach in which teams keep producing valuable software in short cycles and ensure that the software can be reliably released at any time [12] [14]. The CDE workflow aims to assure that the software is deployable throughout its lifecycle and

- *Author name is currently pursuing masters degree program in electric power engineering in University, Country, PH-01123456789. E-mail: author_name@mail.com*
- *Co-Author name is currently pursuing masters degree program in electric power engineering in University, Country, PH-01123456789. E-mail: author_name@mail.com*
(*This information is optional; change it according to your need.*)

that the team prioritizes keeping the software deployable over working on new features [14]. By another word it aims to set-up an environment where developers can commit code to the master branch whenever he or she have completed a task. If there is something wrong with the code the automated tests will fail and the commit will need to be fixed before replacing the latest version of the software which contains a value for the customer [10]. Its scope varies from case to case, and is defined by whatever one needs to do to confidently release and/or deploy the software. This typically involves activities such as code analysis, documentation generation, acceptance testing, regulatory compliance assessments, license scanning and requirement verification [1]. Implementing CDE is challenging [12].
Continuous Delivery (CDE), Continuous Deployment (CD) is a key practice for making software development process reliable and faster. The feedbacks from the Production and Operations team are made available to the developer at frequent stages facilitating improvement and automation. CDE is not a fully automated process. It simply means that the application is potentially capable of being deployed [3]. Continuous Delivery according to software-oriented organizations is when you are able to deliver to the repository when ready, while Continuous Deployment is when you are able to deploy to the target when ready [10].

DevOps is a blend of two terms development and operations, it is considered to be the most effective way to foster collaboration and eliminate the walls of confusion that exist between software developers and operations teams [4]. It is development processes that reduce repetitive tasks in development, quality assurance, and deployment with help of automation tools and work-flows. It improves speed of the delivery, and collection of the user feedback through continuous operation also it reduces development costs, setup time, and failure recovery time through automation [2]. Besides that, it brings new challenges since developers need to be aware of the deployment settings and application runtime characteristics. At the operational stage, several uncertainties, e.g., workload fluctuations and resource availability, may affect the performance analysis [11]. DevOps is all about culture, automation, measurement and sharing (CAMS). It is gaining popularity because of its continuous approach [3]. DevOps is considered as much more of a process, there is not a single tool that helps in deploying DevOps practice in an organization. It is more of a 'toolchain' [5].
As both DevOps and continuous deployment aim at constantly delivering added value to end users, automation is needed to reduce repetitive work [2]. The problems faced before Automatic and continuous integration includes Developer has to wait too long for the test results, and thus wasting a lot of time that could have been well spent in innovation and developing new product, Developer need to go through the entire source code to fix any bug encountered, No continuous feedback from the production or test team at every stage, and Manual configuration can lead to inconsistency, importability and reduced speed of recovery from failure [3][13] it can be handled automatically and the pool of resources can be released immediately after the task is completed [18]. With automated deployments, the risks and downsides of manual deployments can be alleviated, as there is an easy, repeatable and reliable process that maintains documentation of the infrastructure and dependencies in itself [13].
There are five main factors hindering the adoption of DevOps are identified as lack of strategic direction from senior management, lack of education around DevOps, risk of disintermediation of roles, resistance to change, and silo mentality [4].

The goals of DevOps are to form enhancements across all parts in the product and repair delivery they embody improved deployment frequency, quicker time to market, Lower failure rates of new releases, faster recovery time from crashes or failures, and the increased ability to build the right product by fast experimentation [5]. According to 2017 State of DevOps Report, around 24.5% of organizations surveyed resisted automatic deployment due to manually driven rollback mechanism [3].

Continuous Release is a business practice where release candidates evaluated in continuous delivery are frequently and rapidly made generally available to users/customers [1].
Release management is process liable for planning, scheduling, controlling the build, testing and deploying release to increase numbers of successful releases through avoid unexpected outcomes [17]. This process encompasses code change integration, continuous integration, build system specifications, infrastructure-as-code, deployment and release [14].
Release and deployment management goal is to deploy releases into operation and establish effective use of the service so as to deliver value to the client. They additionally ensure handover to service operations takes place which appropriate coaching and documentation exists to confirm ongoing support of the new service. Its scope includes the processes, systems and functions to package, build, test and deploy a release into operation [6]. This means that after a developer commits code to the version control system, it's automatically subjected to a variety of tests and, assuming the tests are passed, placed into production. Modern systems can be deployed multiple times a day [7]. The effective and efficient use of test and staging (pre-production) environments is critical to a successful release deployment [8].
According to 2017 State of DevOps Report, the high-performance companies like Amazon and Netflix deploy thousands of times per day [3].

2 Research Problem

The downtime of the Application is expensive, so most enterprises strive to minimize or, better yet, eliminate it. Reducing downtime when you need to roll out software patches or new features is a technical challenge [9]. The business impact of not being able to bring people, processes, and systems along across development and operations groups is obvious once applications that are the mainstay of a business falter due to

failed deployments and releases. So how does one improve the quality and increase the speed of releases and deployments without compromising environmental stability and control? And how does one streamline processes that span your development and operations teams? Improving and changing the way you release and deploy application software is considered to be a big challenge [8]. As the more frequently you release software, the more reliable your release process becomes, as teams have a greater incentive to fix problems and improve the process [9].
Software is not getting deployed into test environments or released into production Environments any more quickly. Lack of control over the release process, poor collaboration between teams, and manual deployments are all leading to poor quality releases at a high cost to the business. With less than perfect handoffs between development and operations teams, it is not surprising that recent research indicates that through 2016, a lack of effective release management will contribute up to eightieth of production incidents in giant organizations with complicated IT services. Process disconnects between development and operations teams can seriously impact an organization's ability to generate revenue [8].

The amount of required effort to move information and release artifacts from one process to another, or from one team to another, in support of a release is considerable and is a major bottleneck in the flow of the release. Any ambiguities require additional communication between teams to resolve and can result in significant delays, high costs, and failed releases [8]. Also, the pre-release work that isn't automated slows down delivery and makes it hard to roll out new features [9].
We can avoid time-consuming manual work while reducing release risks by following DevOps principles to automate the release and deployment process. Using advanced deployment patterns can help to speed up the software delivery cycle while maintaining control over the way your applications are deployed [9]. As the 'infrequent releases' and 'time consuming to test' were rather common as 36% and 28% of the respondents [18].

3 Literature Review

Kati Kuusinen et al 2018, presents what challenges can a large company face when transitioning towards continuous development and DevOps and how to overcome those challenges, the survey's result was 'infrequent releases' and 'time consuming to test' were rather common as 36% and 28% [18].
Teemu Laukkarinen et al 2018, discusses the fit of DevOps for regulated medical device software development. They examine two related standards, IEC 62304 and IEC 82304-1, for obstacles and benefits of using DevOps for medical device software development. They found these standards to set obstacles for continuous delivery and integration [2].
Aayush Agarwal et al 2018, provides various methodologies and tools that can constitute an effective CD/CI pipeline [3].
Mojtaba Shahin et al 2017, investigating and classifying the factors that may impact on adopting and implementing CD practice [16].
Daniel Stahl et al 2017, highlights a critical problem in the software engineering community by assessing the state of understanding and level of consensus regarding the meaning of continuous practices and DevOps in contemporary literature.it proposes recommendations to improve the level of conceptual clarity in published literature and a set of definitions which offer a way forward to disentangle DevOps and continuous practices from one another [1].
Ahmed Bahaa Farid et al 2017, explained how using practices of DevOps to enhance Lean Software Development that allows to cover the entire life-cycle from development to operations environments. Create a new lean and DevOps framework that used to enhance process of Lean through reduce time to market and increases the rate of software delivery [17].
Morgan B. Kamuto and Josef J. Langerman 2017, the contribution primary to this research is the definition of the main factors that are hindering the adoption of DevOps and a proposed conceptual framework or strategy that can be used to adopt it in large organizations [4].

3.1 Literature review analysis

The maintenance operation of continuous changes is the most effort and cost expensive stage in the SDLC. DevOps has important features that are continuous integration, continuous delivery, automation and high efficiency to increase enterprise market competition [15].
Software companies are increasingly adopting DevOps and continuous software engineering practices to support short feedback loops, gain better control and visibility over deployments, and decrease the need for manual work with the help of automated processes [18].
DevOps is set of practices and principles that is trying to improve life-cycle as a whole through integration between development and operations teams to reduce the release cycles and increase number of software deliveries [17].it relies on automated deployment pipeline that consists of continuous integration (CI), test automation, and release automation. The benefits of the automated pipeline include minimizing manual repetitive work and being able to release whenever working software has been developed which again leads to getting rapid user and customer feedback, i.e. learning fast [18]. And errors can be detected at the time they occur or even there are notifications of an error, this led to reduce the number of days to resolve the errors [17]. These benefits will eventually increase the customer satisfaction, as the customer has a larger and more immediate impact on the product [18].
In a Continuous Integration process, the aim is not to have release ready code, the aim is to integrate often to avoid integration problems, Continuous Deployment aims to deploy the build to a target and Continuous Delivery aims to deliver working software continuously. It doesn't need to include the deployment of software, but it should be ready for a deployment at any given time [10].

Research majority thought that moving towards Continuous Delivery have a positive effect on the development time, also it has positive effects of the quality of the products and that the costs, over time, decreases.
There is an agreement that the short-term costs were high, but

the long-term costs were lower. Higher quality of the product, increasing the efficiency of the work and the possibility to receive and respond quickly to feedback is the greatest benefit of a Continuous Delivery implementation. And that Continuous Delivery is good for their relations with the customers [10].

The deployment pipeline is a staging process, where a set of activities are performed at each step, and evaluated. Allowed to proceed to the next stage. Feedback from each stage is communicated to the developer. This way tracking down a problem is made easier as knowledge about which stage failed is readily available. The purpose of the deployment pipeline is threefold: add visibility, provide feedback and continually deploy [14]. Also, to set up this pipeline was considered to be a great challenge, configuration and setup of the tools are considered to be hard to get working [10].

It can be concluded that automating deployment process also saves developers working time on top of other benefits. The time that was previously spent on preparing for deployments and performing them has been directly transferred into development, testing and support for the end users, enhancing the productivity of the team [13].In general, we can formulate the following statement "small release, small risk" [10].

4 Proposed Technique

Manual releases have been identified as one of the common design problems in software development. Some sources have established that manual deployments are sometimes suitable when the service to be deployed is very simple and is deployed to only few nodes in one or two environments [13].

As it has been previously established from literature review, adopting continuous integration increases the project predictability, developer productivity and communication between teams, this research will present an example of a practical technical improvement to decrease the manual workload of developers and improve the release process using DevOps pipelines by using the following proposed model

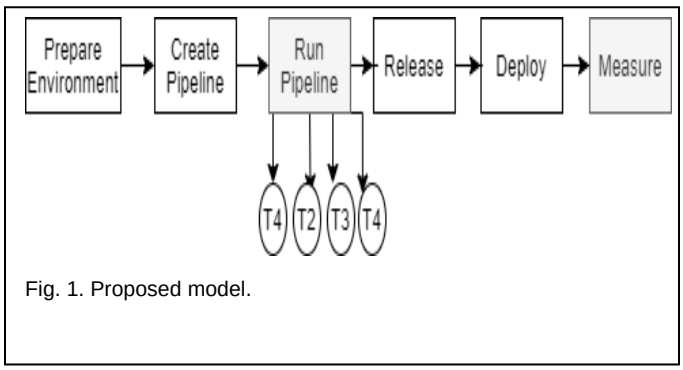

Fig. 1. Proposed model.

At a high level, the pipeline includes compiling, Packaging and running tasks. The first step including preparing the environment to install the agent used to run the pipelines. Second, we will create two pipelines one for builds, and the other for Release also setup configurations and other services. Each pipeline will contain multiple tasks such as (Get Sources, Nu-Get Restore, Build Solution and Publish Artifact that for build pipeline and Copy artifacts to target server, Deploy IIS App, Copy environment configuration and Oracle Communicator Wrapper that for release pipeline). Third, Running the Pipelines to build the pushed code. Fourth, deploy the succeeded build to the staging server. Finally, we will measure Build speed and Deployment speed, Deployment success rate, and Deployment frequency.

5 Conclusion

As explained above in this paper that DevOps enhances the Software Deployment life cycle, enhancing of build and release process are done through identifying the causes of the traditional software deployment wastes and how using DevOps practices in improving and addressing these wastes. The role of DevOps in the addressing or improvement of these wastes reducing time to market and increases the rate of software delivery that leads to lower levels of deployment pains and lower change fail rates.

References

[1] Daniel Stahl, Torvald Martensson and Jan Bosch, Continuous Practices and DevOps: Beyond the Buzz, What Does It All Mean? 978-1-5386-2141-7/17 $31.00, IEEE DOI 10.1109/SEAA.2017.78,2017.

[2] Teemu Laukkarinen, Kati Kuusinen and Tommi Mikkonen (2017 IEEE/ACM), DevOps in Regulated Software Development: Case Medical Devices, 978-1-5386-2675-7/17 $31.00 © 2017 IEEE, DOI 10.1109/ICSE-NIER.2017.20

[3] Aayush Agarwal, Subhash Gupta, Tanupriya Choudhury (ICACCE-2018), Continuous and Integrated Software Development using DevOps, 978-1-5386-4485-0/18/$31.00 ©2018 IEEE

[4] Morgan B. Kamuto and Josef J. Langerman (2017 2nd, IEEE), Factors Inhibiting the Adoption of DevOps in Large Organizations: South African Context, International Conference on Recent Trends in Electronics Information & Communication Technology, May 19-20, 2017, and India,978-1-5090-3704-9/17/$31.00 © 2017 IEEE

[5] Riverstone LLC

[6] Ucisaitil: A guide to release and deployment management

[7] Len Bass, The Software Architect and DevOps, 0740-7459/18/$33.00, IEEE Software, January/February 2018.

[8] Mark Levy, Best Practices in Release and Deployment Management, 162-000088-001/S/08/16/2016 Micro Focu, 2016.

[9] XebiaLabs Enterprise DevOps.

[10] Rickard Bremer and Johan Eriksson, Understandings and Implementations of Continuous Delivery, June 2015.

[11] Catia Trubiani, Weiyi Shang et al, Performance Issues? Hey DevOps, Mind the Uncertainty! , IEEE Software, October 2018, DOI: 10.1109/MS.2018.2875989.

[12] Lianping Chen, Continuous Delivery at Scale: Challenges and Opportunities, ACM ISBN 978-1-4503-5745-6/18/05.2018 ACM/IEEE 4th International Workshop on Rapid Continuous Software Engineering.

[13] Antti Paloposki, Enabling Continuous Integration through deployment automation, Case Study: Property transaction system of Finnish National Land Survey, Aalto University School of Electrical Engineering, January 17, 2018.

[14] Kim Rejstrom, Implementing Continuous Integration in a Small Company, Aalto University School of Electrical Engineering, 2016.

[15] Sen-Tarng Lai and Fang-Yie Leu, A Micro Services Quality Measurement Model for Improving the Efficiency and Quality of DevOps, Springer International Publishing AG, part of Springer Nature 2019, L. Barolli et al. (Eds.): IMIS 2018, AISC 773, pp. 565–575, 2019.

[16] Mojtaba Shahin, Muhammad Ali Babar, Mansooreh Zahedi, Liming Zhu, Beyond Continuous Delivery: An Empirical Investigation of Continuous Deployment Challenges, ACM/IEEE International Symposium on Empirical Software Engineering and Measurement, 2017.

[17] Ahmed Bahaa Farid, Yehia Mostafa Helmy and Mahmoud Mohamed Bahloul, Enhancing Lean Software Development by using DevOps Practices. (IJACSA) International Journal of Advanced Computer Science and Applications, Vol. 8, No.7, 2017.

[18] Kati Kuusinen, Veena Balakumar, Sune Chung Jepsen, Simon Hjortshøj Larsen,Thomas August Lemqvist, Admir Muric, Anna Ølgaar Nielsen, and Oliver Vestergaard, A Large Agile Organization on its Journey towards DevOps, 44th Euromicro Conference on Software Engineering and Advanced Applications. 978-1-5386-7383-6/18/$31.00, IEEE DOI 10.1109/SEAA.2018.00019