IJCSE International Journal of Computer Sciences and Engineering Open Access
Survey Paper Volume-4, Issue-4 E-ISSN: 2347-2693

Survey on Continuous Integration, Deployment and Delivery in Agile and DevOps Practices

K. Sree Poornalinga[1*], P. Rajkumar[2]

[1,2] INFO Institute of Engineering,
Coimbatore, Tamil Nadu, India

Available online at: www.ijcseonline.org

Received: Mar/23/2016 Revised: Apr /03/2016 Accepted: Apr/19/2016 Published: Apr/30/2016

Abstract—Innovations pick up the leap and customers desire quick change, business turning out to be progressively more responsive. Ready end product delivery to market is the solution, and to smooth the progress of the overall business aspiration, software life cycle process needs to be fast. Over the years the transition from waterfall model to agile methodology has come into the era. Progressions of these development operations are moving towards the downstream with the evolution of DevOps. Deployment of software applications in a trustworthy, repeatable, and reliable approach meet up the demands of an agile development which can only be completely achieved by embracing automation. Several DevOps main beliefs are supported by Amazon Web Services (AWS) which every IT departments can profit fromand thus business agility is improved. In this paper, we focus on delivering the principles of DevOps and Continuous Integration, Deployment and Delivery practices supported by them.

***Keywords*—**Amazon Web Service, Continuous Integration,Continuous Deployment, Continuous Delivery, DevOps, Software Life Cycle.

I. INTRODUCTION

Earlier days in the software industry, one of the most challenging and tense moment during project development phase was integration. By last few years, integration has largely disappeared as a source of pain for developers, lessening to a non-event. The principle of this revolution is the practice of integrating the code often and continuously.

Continuous integration (CI) is a software engineering practice which reduces the blind spots of software development and leads to the process of building and delivering the software in rapid. At first, a daily build existed as standard. At present, each team member has to submit their work on daily basis and intended for a build to be accompanied with each significant alteration. When CI used by the book, provides constant feedback on the status of the product development. For the reason that CI discovers short ages initially during development process, defects are normally less significant, less complex and stress-free to resolve.

The continuous integration, deployment and delivery is inevitably multifaceted, across the development, testing, staging and production environments. This run-through assist the software development team in evading or noticing compatibility glitches in prior. CIDD is a development practice that calls for developers to integrate their own code keen on a shared repository more than a few times a day. Every single check-in is then tested by an automated build, letting developing teams to become aware of every problems so often and earlier.

CIDD is considered to be one of the best practice in supporting software development team throughout the production environment. “Continuous Integration doesn’t get liberatedfrom bugs, however it makes them radically easier to discover and eliminate.” On the other hand, Continuous Deployment is closely associated with Continuous Integration which bring up the process of deploying the product code across different software development environment that reduces the time from days to hours and hours to minutes. Also, Continuous delivery is another practice that dictates the fast delivery of the product to the production line. Essentially, “it is the practice of releasing every good build to users,” as in [3]. Organization by embracing the Continuous Integration, Deployment and Deployment, it not only lessens the risks and catches the bugs quickly, but then also travel quickly towards the working product. By means of low-risk releases, one can promptly turn out to be comfortable to business necessities and user requirements.

This agree to for better team work between operation and delivery, operating real change in every single software organization, and spinning product release practice into a greater advantage. As Continuous Integration, Deployment and Delivery (CIDD) in manual is a messy process, i.e. error prone. AUTOMATION is the key to cutting this task down to size; it enables the team to help deliver code faster. Number of tools for Automating CIDD practice is available in the market using which every organization should adapt for continuous product integration, deployment and delivery. But due to the lack of knowledge in CIDD tools and its new environment, the value of these practices is not widely recognized. We propose to dig the importance of the

CIDD adoption for Agile and DevOps culture. For which we tend to demonstrate by creating a newly structured infrastructure called "CICI" that automates the Continuous Integration (CI) practice in a Cloud Infrastructure (CI) for the improvement of software quality and productivity. This infrastructure provides a complete environment for the developing team to build their product in an efficient manner.

II. LITERATURE REVIEW

DevOps is a growing term that chiefly spotlights on healthier teamwork, communication, and incorporation sandwiched between software developers and IT operators. Formerly, many organizations works with broken infrastructure for integrating the code end to end throughout the software development life cycle.Principally, developers are likely to build software and make frequent changes to it, whereas the focal point of IT operators is on maintaining software stability and reliability. This divergence between the goals of two teams lead to inconsistency, and in the end the business process suffers. DevOps ensures the quality of the product at all stages as in [4]. This can be envisioned as a conveyor belt, where lots of check-ins and stability are happening in one place, at every stage, in order to ensure that any package approaching down the belt is detached if it's not high-quality as much as necessary, and finally delivered to the production environment safe and sound. Any practices that is adopted by the organization which speeds up the business process in the form of cultural change, is otherwise termed to be DevOps. The automation of Continuous integration, Deployment and Delivery (CIDD) are such practices that help in bringing the ideas of DevOps as practical solutions.

Continuous Integration (CI) is considered to be one of the best software development practices where members of a team integrate their work frequently; commonly each person integrates at least daily leading to multiple integrations per day. The integrated code is then verified with the help of automated build (including test) in order to detect the integration errors as quickly as possible. A lot of teams finds this approach leading to significantly reduced integration glitches and permits a team to develop organized software more promptly. Martin Fowler et al, provides a fast outline of Continuous Integration summarizing the technique and its existing usage in enterprise world [8]. The author focused on describing the key practices in making up an effective Continuous Integration (CI). As a result, developing a well-organized and automated building process is crucial in a controlled environment. Various software experts say that CI has numerous benefits, but we've found that it's still a rarity in the IT field. Therefore, the key solution is to automate the CI practice absolutely everything and run the development and testing process so often, in that way all integration errors are found quickly and early. As a result each individual is equipped for the transformation of things whenever they needed, because they know that if they do grounds an integration error, which is quite easy to catch and hit, as in [13] and [14].

Continuous integration (CI) is the key element involved in supporting agile software development mainly in testing environment [12]. Sean Stolberg projects his understanding in implementing and supporting continuous integration contained by the framework of agile methodology [9]. The transition of traditional software tester to an agile development environment gives a clear viewpoint that there needs to put the indispensable infrastructure in place and promoting an improved development practices in order to make the changeover to towards agility. It is noted that continuous integration implementation is the solution to the lack of automation framework. And also the organization that follows agile practice cannot find success without continuous integration. Despite being a cornerstone of agile development, surprisingly little is known about how organizations assimilate continuous integration (CI) and what organizational changes adopt to that practice.

All aspects of DevOps applicable to various phases of SDLC are addressed in [3]. And it specifically talks about the business needs, ways to move from continuous integration to continuous delivery and its benefits. As part of Agile transformation in past few years we have seen IT organizations adopting continuous integration principles in their software delivery lifecycle, which has improved the efficiency of development teams, as in [12]. With the time it has been realized that the optimization of continuous integration alone is just not helping to make the entire delivery lifecycle efficient or is not driving the organization efficiency. Unless all the pieces of a software delivery lifecycle work like a well-oiled machine - efficiency of organization to optimize the delivery lifecycle cannot be met. This is the one of the main problem which DevOps tries to address. The journey and learning process in setting up a Continuous Integration for software group [1]. Analysis shows that the success of continuous integration is very much depends on the tools selected and discipline of the team. The value of an integrated environments, modernize the build process where any software engineers could learn or adopt immediately, all these needs lead us to the idea of Continuous Integration. Continuous Integration is not necessary for the one who works in isolation but collaborates with other important task involved in software development life cycle, as in [7].

Almost all software development is performed by means of teamwork, leveraging on varied functional groups carrying out different components or subsystem. In an enterprise where development of software involves a collection of developers working on modules, integration management is absolutely a necessity; we need to find ways to labour proficiently and effectively in making the lengthy and substantial integration practice in to a modest and delighted job. Whether the practice of Continuous Integration of agile software development methods has had an impact on open source software projects has been investigated in [2]. Commercial software firms are increasingly using and contributing to open source software. Thus, they need to understand and work with open source software development processes. Using fine-granular data from more than 5000 active open source software projects the analyses of the code size contributions over a project's life-span. Code contribution size has stayed flat. It has been inferred that the open source software development has not reformed their code integration practices. In particular, within the limits of this study, it is claimed that the practice of continuous integration has not yet significantly influenced the behaviour of open source software developers.

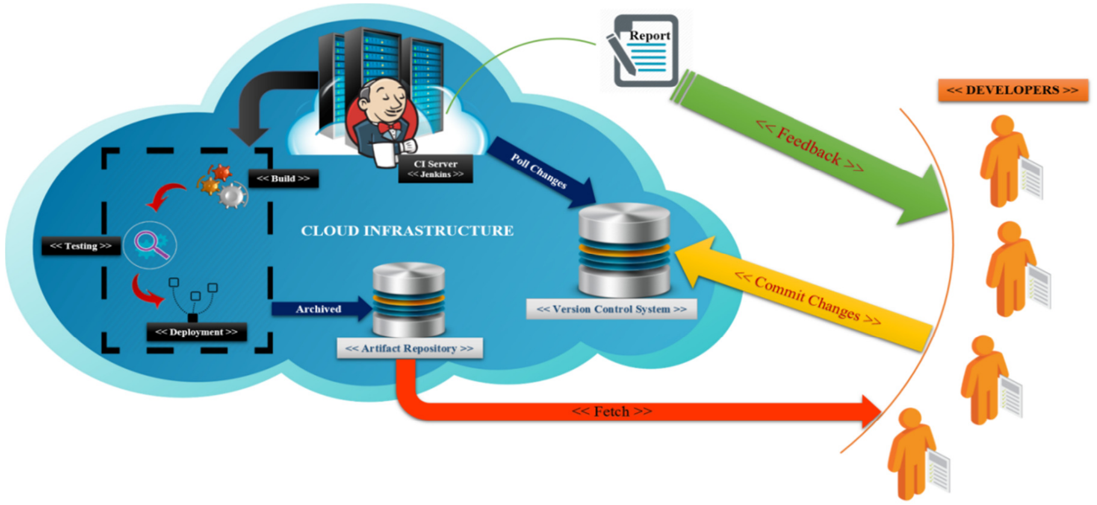

Figure 1: Continuous Integration (CI) in CIDD AWS Infrastructure

Organizations faced challenges while adapting to Continuous Deployment has been examined as well as the approaches to alleviate those challenges are studied [5]. Continuous Deployment (CD) is added emerging software development process which was positively applied by the number of core companies such as Facebook, Microsoft, Google and IBM. The CD process aims to immediately deploy software to customers as soon as new code is developed, and can result in a number of benefits for organizations, such as: new business opportunities, reduced risk for each release, and prevent development of wasted software, as in [6]. The author set up a total number of 20 procedural and community interviews deriving the challenges faced by organizations while facing to adopt continuous deployment process. And so, it has been determined that every organizations who needs to be adopted for CD practice has to be well prepared to knob technical as well as social challenges while adopting with their existing expertise. The researchers interrogated 15 information and communications technology companies to govern the extent to which the organization takes on continuous deployment [3]. It has been found that continuous deployment is beneficial and there lies hindrances during its adoption. In spite of understanding delivered benefits, none of the software companies adopted a completely automated deployment pipeline. The companies also had advanced CD capability than what they have accomplished. In many cases, organization intentionally chooses to not to aim for full continuous deployment.

Continuous integration, Deployment and Delivery (CIDD) has been everywhere for a while now, but the habits it suggests are far from common practice. Automated builds, a full test suite, and committing to the mainline branch every day sound simple at first, but they require a responsible team to implement and continuous care. CIDD starts with enriched tooling which can be catalysed for lifelong change in every software organization's culture. Fazreil Amreen Abdul et al illustrate the journey and learning process in setting up a continuous integration, deployment and delivery for a software group, as in [5]. The analysis report shows that the success of CIDD solely very much depends on the tools selected for automation and discipline of the every single individual/team. Discipline of the team means most of the time the human (managers, programmers and developers) state of mind habits the abilities on achieving the organizations objective (e.g. money first, quality later) which stop CIDD from being the part of their practice.

Analysis shows that the implementation of Continuous integration effectively eliminated the need for integration testing and the cost associated with developers spending time on this phase. The philosophy of CI which on a regular basis integrates the changes of individual developer's code with the mainline base set aside the whole development team from integration hell that has been coined from extreme programming. The practice CI is reinforced with the help of automated tools to cope with repeated integration of source code through automated builds, testing, and deployments. Some of the obtainable products from the global markets are, for example, Jenkins/Hudson, Assembla, SonarQube or GitHub, allowing for the execution of a unified CI-process, as in [10]. One of the main problems, however, is that relevant information about the quality and health of a software system is both scattered across those tools and across multiple views. One of the challenging problem by raising awareness on software quality and tailoring material towards individual stakeholders, such as developers, team lead/testers, as in [9]. On behalf of this reason they contemporary a quality awareness charter and stand named as SQA-Mashup, as in [11]. It marks the use of service-based mash up standard and integrates the information starting from the complete CI-tool sequence in a solitary provision.

III. PROPOSED SYSTEM

The objective of this paper is to dig the importance and advantages of DevOps and make the complete transformation of manual software development process to automated system through Continuous Integration, Deployment and Deliver practices. Therefore, we propose to create new Infrastructure in Amazon Web Service which automates the complete pipeline for CIDD. The new infrastructure automates every process from build in development to delivery in production line. With the available technologies in the global market, selective CIDD tools are used for the implementation of the proposed infrastructure.

The automation system consists of the following five stages:

a. Source Code Analysis
b. Automate Build (CI)
c. Automate Testing (CI)
d. Automated Deployment
e. Automated Delivery

CIDD provides wide-ranging profitable benefits to software engineering world. On the ground, the profits of Integration, Deployment and Delivery (CIDD) are: Suppression of manual software deployment, Prevention production errors, reduction of staging and providing report analysis on code quality. In business terms, the value of Continuous Integration is: Reduction of risk, decreased overheads across the development, testing and deployment process also, improving the standing of the business by providing improved Quality Assurance.

Organization adapted to development life cycle without automated CIDD face lots of bugs, infrequent commits, difficulty while integrating the code, insufficient testing, slow release process, poor project visibility, high maintenance cost, inflexible code bases which altogether leads to unhappy clients. Whereas CIDD Infrastructure provides the automation of build, test code quality metrics, etc. when organization adapts to such practice, it offers a dedicated build server, regular commits, fewer bugs by means of early frequent testing and regular automated release. The automation provides smoother integration process, automated regression tests and better visibility.

IV. CONCLUSION

According to our study, none of the organizations in has reached the *infusion* stage in automating Continuous Integration, Deployment and Delivery (CIDD) for adapting to DevOps culture. Despite of its own advantages to the business development process; only less than 65% of the organization implements CIDD in manual, and very few companies automated partially or fully to the CIDD practice.

It is clear that the importance of CIDD is not recognized widely due to the lack of labours and knowledge of tools and environment. In order to help the development team in software engineering world, we propose to create a newly structured Infrastructure that simply automates the CIDD practice in AWS Cloud environment for the improvement of both software quality and business productivity. The CIDD AWS Infrastructure automates the product build, testing and deployment from development to production line in a cloud environment for the productivity of developing team.

REFERENCE

[1] Alexander Eck, Falk Uebernickel, and Walter Brenner, “Fit For Continuous Integration: How Organizations Assimilate An Agile Practice,” 2014.

[2] Amit Deshpande and Dirk Riehle, “Continuous Integration in Open Source Software Development,” 2008.

[3] Daniel Ståhl and Jan Bosch, “Experienced Benefits of Continuous Integration in Industry Software Product Development: A Case Study,” 2015.

[4] David Chapman, “Introduction to DevOps on AWS”, Amazon Web Service, December 2014.

[5] FazreilAmreen Abdul and MenselyCheahSiowFhang, “Implementing Continuous Integration towards Rapid Application Development,” May, 2012.

[6] Gerry Claps, Richard BerntssonSvensson, and Aybüke Aurum, “On the Journey to Continuous Deployment: Technical and Social Challenges Along the Way,” 2014.

[7] Hanna Salopaasi, “The Role of Continuous Integration in Software Business,” 2014.

[8] Manish Virmani, “Understanding DevOps & Bridging The Gap From Continuous Integration To Continuous Delivery,” 2015.

[9] Martin Fowler, “Continuous Integration,” 2006. [Online]. Available: http://www.martinfowler.com/articles/continuousintegration.html

[10] Mathias Meyer, “Continuous Integration and Its Tools”, IEEE Software, 2014.

[11] Martin Brandtner, Emanuel Giger and Harald Gall “SQA-Mashup: A Mashup Framework for Continuous Integration,” Information and Software Technology 65, October 2014.

[12] Sean Stolberg, “Enabling Agile Testing Through Continuous Integration,” 2009.

[13] K. ReshmaRevathi, Dr. S. Kirubakaran, "A Survey on Automatic Bug Triage Using Data Mining Concepts", International Journal of Science and Research (IJSR), ijsr.net, Volume 5 Issue 3, March 2016, 184 - 186

[14] D. Cubranic and G. C. Murphy, “Automatic bug triage using text categorization,” in Proc. 16th Int. Conf. Softw. Eng. Knowl. Eng., Jun. 2004, pp. 92–97.