A Survey of Continuous Integration, Continuous Delivery and Continuous Deployment

Sergejs Bobrovskis, Aleksejs Jurenoks[1]
[1] Institute of Applied Computer Systems, Riga Technical University, Latvia

Abstract. Modern Information Technology (IT) practices are heavily dependent on the organizations' ability moving changes/features/fixes live in an agile and errorless way. As technology evolves, the requirement to implement automation practices increases exponentially. Over the past decades, waterfall software development practices are substituted by agile methods of software development. This shift undoubtedly brings businesses, operations, and software developers closer to each other towards the commonly established goals. The market introduced a great amount of open-source and proprietary solutions to facilitate this merger. In turn, technologies itself also augmented, and as a result, new practices have emerged, namely continuous integration (CI), continuous delivery (CDE) and continuous deployment (CD), all supported by cited open-source and proprietary solutions. This research represents an initial step towards designing and developing methods for automated distribution of resources in a programmable environment, by doing market research and analyzing related articles to verify and summarize the current status of CI/CDE/CD.
Keywords: Continuous integration, proprietary solutions, DevOps, IaaS, PaaS, SaaS, IaaC, GIT.

1 Introduction

Information Technology has closely integrated into all industries and domains (for-profit and non-profit organizations), goods delivery automation to consumers plays a crucial role in “surviving” on the marketplace.

Historically organizations had to maintain a manual quality assurance (QA) teams represented the last security layer before any updates/changes could go live (this is still valid for many organizations). It is worth to mention that several decades ago programming languages started to evolve exponentially and the broad variety of new and derivative languages appeared on the market. Depending on the application languages differ from case-to-case, including database (DB) solutions (SQL vs. NoSQL), frameworks (for example, flask, Django, Node, and others), iOS, Android, Web front-end/back-end and automatization practices (with a custom syntax).

Shifting from waterfall to agile practices QA teams without automation procedures are deemed to becoming "an outsider." It is critical for QA teams to apply automation methods and tools within a dynamic environment by the integration of CI/CDE/CD in order to stay in the market and proceed with the established business goals. [1]

This article tends to perform literature source research review and analysis of developed and established practices amongst researches, communities, and blogs for CI/CDE/CD in order to find known and undocumented flaws.

The further structure of the article is as follows:

Chapter II focuses on explaining the differences between CI/CDE/CD and explaining the involvement of cloud-computing in IT industry, giving the overview of pros and cons of various existing solutions on the market.

Chapter III is a content analysis of related articles with emphasis on finding a potential solution for the research subject, in order to further expand and clarify previously undocumented issues.

Chapter IV addresses methods for automated resource distribution in a programmable environment.

Chapter V concludes the findings and explains the steps required and set future research focal points.

2 Continuous Integration, Continuous Delivery, Continuous Deployment Market Distribution, and Practices Research

There are several market-accepted automation practices:

- CI – is the practice that instructs developers to commit changes several times per day into common repository/mainline. [2]
- CDE – is the practice that tends to ensure every available build is stable, checked and verified to be pushed live on a single button click. [3]
- CD – is the practice that further expands CDE by ensuring the build is not just ready and checked, but automatically get pushed live onto production environment so real consumers could see changes delivered immediately. [4]

All aforementioned practices are interrelated and are the consequence of the desire for automation striving toward consuming more market place by allowing rapid and errorless deployments.

In order to move forward CI/CDE/CD practices, organizations had to move from old-fashion on premise server infrastructure to cloud-based solutions. In turn, this has led to the appearance of new practices, such as:

- Software as a Service (SaaS);
- Platform as a Service (PaaS);
- Infrastructure as a Service (IaaS);
- Infrastructure as a Code (IaaC);
- Network as a Service (NaaS);
- Communication as a Service (CaaS).

A further outcome of the development of information technologies are [5]:

- Microservices – divides technology stack and development of features into separate services.

- Containers – ready-to-work application logic encapsulated to so-called packages, that can be distributed anywhere, be portable, and should perform programmable actions. Containers are not microservices; however, microservices can be integrated into containers;
- Docker – is an orchestration of containers, having an internal architecture and management tools.
- Kubernetes – is a current trend and provides a container-centric management environment.

The theory mentioned above is critical to be aware of, as CI/CDE/CD solutions, one way or another, are connected to cloud computing and community-accepted orchestration tools for reaching company goals and sustaining on the market with rapid and errorless deployments.

3 Related Research Works

Reading through academic and research articles, the author sees CI/CDE/CD are relevant thematic as organizations keep searching for the lightweight solutions to stay "alive" in the business.

Most of the articles read mainly considered explanations of various solutions open-source and proprietary that exist on the market and accepted by various online communities. Some authors are trying to dig into specific CI or CDE proprietary tools, giving critical explanations for the implementation phases (which the author believes as an essential aspect, alleviating the resolution for making a decision). [6]

Moreover, the author has found several papers focusing on the challenges organizations are facing during the integration of CI/CDE/CD practices. Few examples are changing the culture of an organization, qualification expansion of developers and operations team, "cracking" the business, so business truly believe spending money will impact on return on investment (RoI) [7]

Main observation authors noted that various articles focus on deployment procedure itself, and explaining prerequisites and challenges organizations need to go through. Specifically designing deployment pipelines, using described tools. Important to note that because of the huge amount of programming languages, a unified solution does not exist; thus main focus is explaining technological stack with further explanations of its use. The author believes it is an important consideration, as organizations might refer to the paper and its findings and make appropriate configuration of their infrastructure. The paper "Continuous Integration, Delivery, and Deployment: A Systematic Review on Approaches, Tools, Challenges, and Practices" does a great job in explaining tools, approaches, steps for implementation of overall automation, it covers most of the market-wise open-source and proprietary solutions.

One paper from the author's point of view deserves a separate attention "Chuck Rossi, Kent Beck, Elisa Shibley, Tony Savor, Shi Su, Michael Stumm "Continuous Deployment of Mobile Software at Facebook (Showcase);" author has been surprised that Facebook assisted and allowed to write a paper describing the actual development procedures within the company. It only touches aspects for mobile devices, but the

deployment system Facebook has built is incredible. Manipulation with the source code, commits security checks, separate rooms with cameras focusing on analyses of updates on the piloting of changes, feature-toggles, fully automated continuous delivery, a colossal amount of tests constructed with one secure pipeline and many other practices. The author truly believes CI/CDE/CD is not a simple application of tools, but a tremendous challenge of changing the overall approach for development, ensures developers and operations start to think differently, truly changing the way-of-thinking rather than the tools used. [8]

Another few papers are focusing mainly on testing procedures, as the author states quality and security can be reached solely by passing all automatic tests, including applying security conformity tests. At some point, organizations can refer to specifically designed security practices. The Software Assurance Marketplace (SWAMP) was established to support Continuous Assurance (CA). The Author explains the tools to be used, steps to execute, and empirically prove the concept to ensure created deployment package can be considered secure for being pushed further into live environments. [9]

Interestingly the author found only one paper focusing on automation. Mostly this is an explanation and enhancement of the tool called "GuideAutomator." Even though this is not just a GUI tool, it calls for having a programming experience in writing Macros, but it allows going through the code, executes a group of tests and creates visual documentation directly from the source code. Of course, developers have to follow a specific procedure how the code has to be written, commented and follow other criteria. This again touches aspects of changing the overall organizational culture. It is even more critical nowadays, as in the last ten years Agile methodology consumed most of the marketplace. Some companies use it in a discreet mode, and others are "hiding" behind Agile principles in order to prove the customer that they are focusing on delivering goods and services, rather than following old-fashion Waterfall solution for ample documentation. However, reading carefully, the Agile methodology does support documentation, but less considerably. Nonetheless, the documentation has to be done. Otherwise, at a later time, an organization could face issues that might lead to company closure

One of the empirical studies the author liked, is the example of how companies misuse configuration for CI/CDE/CD. The author wants to make it as a separate paragraph, as realizing how big the threat and impact of misuse of deployment pipeline can cause, as not knowing what going to happen and why a company might become wrecked by launching a deployment pipeline that has flaws. Thus by choosing deployment configuration tools, it is worth checking whether it offers analysis tools for checking the syntax for overall security and misuse configuration.

Other papers read by the Author focus on overall explanations of benefits and challenges organizations will encounter during the integration of CI/CDE/CD. Overall tools and its explanations, programming languages and orchestration tools with association with cloud computing. Papers mentioned above are worth to read as overview papers.

4 Overview of Methods and Tools for Automated Distribution of Resources in a Programmable Environment.

4.1 Topicality of automated distribution tools

As the author stated earlier, we are living in a dynamic and rapidly developing world. Direct expansion of IT systems into all aspects of our lives, especially when the Internet becomes available in our pockets, with the launch of the first iPhone by Steve Jobs, dated January 9, 2007. Organizations realized to win the market competition they have to deliver stable, errorless and secure applications. Even though mobile browsers exceeded the usage of the desktop, this concept is still valid. However, timewise users spend more time browsing the web with desktop browsers than mobile devices. The ability to download, and deliver applications and new features almost on the fly, gives an excellent opportunity for automation implementation into all aspects of development and deployment procedures. With the further introduction of 2G,3G,4G,5G mobile internet technologies, and low prices for the usage gives additional confidence more and more users would use applications on mobile devices and desktop devices. Logically, they have to be in sync all the time and have the same amount of feature set introduced across mobile and desktop environments to stay competitive. Such development can be achieved with an automatic and concurrent release across all environments and proper planning.

Having lots of programming languages and tools, organizations doing startups tends to apply community proved approaches, so-called best-practices, associated programming languages (where the support is high), and orchestration tools for delivering the product in best possible shape.

Worth to mention, lots of organization going to go through a transformation of their infrastructure and development culture to meet expectations of a user. Some organizations have heavily invested in enterprise solutions, which makes making changes for them to be a pain. Since the Author is currently directly involved in a transition from an on premise infrastructure storing application towards a cloud solution, the Author is fully aware of difficulties development, operations, testing and business teams are facing. Companies spend thousands of euros in trying a different proof of concepts (PoC) before a proper solution can be chosen. Moreover, a third party organization for penetration testing, quality assurance, security, and business logic been involved, verifying all aspects found by PoC. Various metrics, thresholds set-up to traverse through the whole process, verifying that real users will not suffer. Special software architecture team with common strength are developing a secure pipeline for CI/CDE. CD is not used as organizations having Enterprise solutions cannot allow CD in full (as does the organization the author currently works at), as additional approval steps must always be preserved until all parties involved put signoff and commit on changes that can be pushed live. In that case difference between CDE and CD is just a button click, everything else operates in an unattended mode.

Thus, best practices having CI/CDE/CD in place calls for architectural changes, cultural changes, various PoC for approving the direction, and most critical calcula-

tion of benefits a company will get. In most cases, it consists of cost management, easy-to-use, and support solutions, more rapid deployments on delivering goods (ROI value get raised), ability to monitor the whole infrastructure through one Dashboard and other related actualities.

I/CDE/CD is continuously growing, and current market-trend shows it will not ever stop emerging.

4.2 Methods of automated distribution tools

The author realized, since there is none universal approach or methods for developing CI/CDE/CD within a dynamic environment, most of the researches use literature review method or empirical study research methodology.

Upon defining organizations' infrastructure, security, technological requirements, using methods above can choose the best approach and tools matching organizations current needs.

Literature Review Method is focusing on analyzing a massive amount of articles, guides, best practices, community reviews, company success stories, official tools documentation with an aim to find correlation path between requirements, approaches and available tools.

Most of the authors highlight specific technological stack before they do an in-depth literature review, namely programming languages, cloud computing solution (cloud, on-premises, and hybrid), database, enterprise or non-enterprise solutions, desktop or mobile or hybrid applications, static or dynamic environment. These influences on the literature to be chosen for analysis.

In its turn, the empirical study research method also bounded with a technological stack, however mostly focusing on execution of the actual solutions.

Articles using empirical study tend to divide the paper into logical interrelated modules, such as Business Requirements High-Level Overview
Based on company decision for these items, various tools can be considered further.
IT Orchestration tool allows to visualize and automate routine tasks associated with CI/CD or plays a role for a mediator for different aspects, like cloning the initial set-up, instead of writing from scratch whole pipeline.

These are tools allowing to compile a build (again depends on the language used), create a distribution or ready-to-publish package for iOS/Android.

The author mostly prefers the second way as an empirical study, as currently representing both academic vs. practical approaches, thus reading through articles or even better Master's Thesis of other authors can be easily chosen and applied the described approaches for currently in charge projects.

4.3 Resources Distribution Automation Tools

From the practical standpoint, the author has been involved in an implementation of several CI/CDE/CD solutions within a various technological stack. Thus, the author would like to focus on several solutions directly involved with the mentioned above.

Since projects are using different preprocessors, like Sass or Less (derivation for CSS) or CoffeeScript (derivation for JavaScript), Haml (derivation for HTML), various frameworks (like Flask or Django), additional steps needed for compilation. Specific deployment pipeline has been introduced to combine various steps in order to create an end-package.

To reduce the amount of information overall to process the solution looks as follows:

- Jenkins works with specific GIT branch, and creates/compile a distribution package;
- Jenkins triggers unit/functional/operations testing and send out notification upon failure;
- In order to address security requirements, Jenkins build several Debian (.DEB) packages,

This was necessary to avoid developers seeing all access keys within a code for the Prod environment. Also, used for environmental separation (Prod, QA, Dev), using least-privilege approach.

This allowed to develop and test independently, dedicating access only to those individuals who should have the rights and permissions.

- Ansible launching runbooks can connect from a master server through SSH all to other servers in a cluster and execute “dpkg –I” command for installing three compiled packages;
- To ensure we have zero-time deployment, first step been taking out servers from balancer (to stop serving user requests), apply changes, restart the server and put it back in service.

This is a high-level overview of the procedure we used (it has additional steps (association with third-party services for security review), based on signed NDA not allowed to give explanations.

In the author's second project technological stack is different and not everything fully completed and is in the works.

With the introduction of Microsoft Azure, a special orchestration tool for CI/CDE/CD become available, namely VSTS.

These points are a fraction of other capabilities VSTS offers. However, interesting to note, that one tool combines CI/CDE/CD, depending on the execution path one plans to follow.

Similar to the first project, here we are using mostly Java and .NET languages, which also calls for compilation procedure.

When the compiled package (.WAR / .JAR) is ready for distribution, the VSTS deployment pipeline can have numerous steps for delivering changes/updates to the client.

Service Endpoint mentioned above is having the rights to manipulate with artifacts within Microsoft Azure resource groups. Resource groups are a combination of arti-

facts associated with one “commonplace,” it allows to set up rights and permissions, as well as delete everything in one button click.

Depending on the Microsoft Azure language support, the author believes Microsoft is tending to reach technology less environment for being able using any programming languages within a project and VSTS as orchestration tool.

Theoretically, one can use Microsoft Visual Studios (latest versions if possible) as well, it also has individual modules to work with Azure, and thus allowing you to write code as using IDE with extension modules, and execute tests and do deployment write from Microsoft Visual Studio. Everything depends on the practice organization would like to follow.

5 Conclusions

Looking through articles, available tools, CI/CDE/CD practices, and available methods author proved universal out-of-the-box approach does not exist. Exceptional cases, when organizations do work within the proprietary environments, like Microsoft Azure and use dedicated tools for orchestration, namely VSTS. This approach leads to confinement, as the inability to combine technology might lead to showstopper in the future development cycle.

Also, looking through available tools, both open-source & proprietary solutions, they all using different languages, have its flaws, and lack of maturity. Lack of maturity is a critical case, as one tool quiet rarely can satisfy all of the organization's demands and requirements. Thus most organizations have to allow using more than one tool.

Usually, Enterprise-level organizations have special procedures for security certification of tools or agents that can be requested from the self-service portal when certification is passed. However, this process (to the author best knowledge and experience) lasts from two to three months, which is again a subject to seeing new updates/upgrades of the feature when certification is finished and approved, again start re-certification procedure in case this functionality is required.

To address all concerns raised, the author plans to develop a software delivery framework focusing on CI/CDE/CD consisting out of dynamic modules, where different-level (small, mid, big) organizations might take "templates" and available tools (with explicit explanations) and easily tailor (construct) own CI/CDE/CD solutions having full explanations "on the table".

There are various maturity models, explaining steps to take in order to build CI/CDE/CD, but none offer explanations allowing companies to make proper and weighted decisions. In most cases, a list of tools explaining Pros & Cons.

Also, planning to develop a module-based method. This module will allow substitution of other modules. This will, in turn, allow the dynamic addition and subtraction of steps within a module. Such flow will not sabotage the entire CI/CDE/CD flow. This is an exceptional case, for when companies do decide to change their instruments for CI/CDE/CD, but in case of, like if support of the tool is stopped or support made critical updates which impacted on the inability of using it further.

References

1. Youssef Bassil. “A Simulation Model for the Waterfall Software Development Life Cycle”, International Journal of Engineering & Technology (iJET), ISSN: 2049-3444, Vol. 2, No. 5, 2012 http://iet-journals.org/archive/2012/may_vol_2_no_5/255895133318216.pdf
2. D. Stahl and J.Bosch. “Modelling continuous integration practice differences in industry software developent”, Jo urnal of Systems and Software, vol. 87, pp. 48-59, 2014.
3. J. Humble and D. Farley, Continuous Delivery: Reliable Software Releases Through Build, Test, and Deployment Automation. Addison-Wesley, 2010.
4. C. Caum. “Continuous Delivery Vs. Continuous Deployment: What’s the Diff?” https://puppetlabs.com/blog/continuous-delivery-vs-continuous-deployment-whats-diff, retrieved January 20, 2016.
5. Viktor Farcic. DevOps 2.0 Toolkit “Automating the Continuous deployment Pipeline with Containerized Microservices”
6. Aleksi Hakli. "Implementation of Continuous Delivery Systems", Faculty Council of the Faculty of Computer and Electrical Engineering of 4 May 2016
7. Mojtaba Shahin, Muhammad Ali Babar, Mansooreh Zahedi, Liming Zhu "Beyond Continuous Delivery: An Emmpirical Investigation of Continuos Deplpoyment Challenges" In Proceedings of 11th International Symposium of Empirical Software Engineering and Measurement (ESEM), Toronto, Canada.
8. Chuck Rossi, Kent Beck, Elisa Shibley, Tony Savor, Shi Su, Michael Stumm "Continuos Deployment of Mobile Software at Facebook (Showcase)
9. Zeba Khanam, Mohammed Najeb Ahsan "Evaluating the Effectiveness of Test Driven Development: Advantages and Pitfalls"