WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

Software Automation Enhancement through the Implementation of DevOps

Sarthak Srivastava[1], Karthik Allam[2], Anirudh Mustyala[3]

[1]Visa Inc, United States
[2]JP Morgan Chase & Co, United States
[3]JP Morgan Chase & Co, United States
DOI: https://doi.org/10.55248/gengpi.4.623.45947

A B S T R A C T

The DevOps paradigm for developing software unifies the activities of development, operations, and quality control. The primary objective of DevOps is to streamline the software development process through automation, while also emphasizing cloud security. DevOps gives a scope of instruments, like Git, Jenkins, Chef, and Selenium, to empower automation all through the whole software development lifecycle, including planning, building, integration, testing, and deploying. Microsoft Cloud Azure and Visual Studio Team Services (VSTS) are commonly used platforms for DevOps. By facilitating continuous integration, continuous delivery, continuous testing, and release management, DevOps enables organizations to save time and resources while minimizing manual effort. Furthermore, real-time activity tracking by project participants allows them to bridge the gap between continuous integration and continuous delivery.

Keywords: TFS, VSTS, IIS, Release Management, Continuous Integration, Visual Studio, Triggers, Build and Release, DevOps, Software Development Lifecycle.

Introduction

DevOps emerged as a movement and philosophy approximately 2-3 years ago with the goal of improving collaboration and communication between development and operations teams and building trust. DevOps is a software development approach that aims to unify the development, operations, and quality assurance processes while strongly advocating for automation. It combines development and quality assurance, encompassing requirement enhancement, request, and bug fixing at the design stage. Customer feedback is solicited after deploying the software product, allowing the deployment team to make changes to the product and deployment process. This iterative approach enables the team to improve both the product and the deployment process by reducing rework and process overhead.

Abbreviations and Acronyms

VSTS: With the help of specialized tools and roles for software architects, developers, and testers, the Microsoft Visual Studio extension VSTS enhances Visual Studio's abilities to support development teams. For the requirements of application developers, it offers a range of hosted DevOps services. TFS: Team Foundation Server (TFS) is a Microsoft offering that offers source code management capabilities, either through Team Foundation Version Control or Git, as well as features for reporting, requirements management, project management, automated builds, and lab management. The on-premises version of VSTS, or TFS, enables installation and management on a private server.

IIS: To host websites and other web material, Microsoft created Internet Information Services (IIS), an extensible web server, to provide requested HTML pages or files.

Release Management: Release Management refers to the arrangement of methodology that regulate the preparation, planning, and control of the build, test, and deployment of releases, with the goal of routinely creating and deploying software applications across various platforms.

Continuous integration: The practice of Continuous Integration (CI) in software development requires developers to frequently integrate code into a shared repository several times per day. Automated builds then verify each check-in, allowing teams to quickly identify any problems and take necessary actions.

Builds and Releases: An application that has been produced by the development team and submitted to the product testing team is alluded to as a "build" in the software product development industry

Description:

DevOps is a software engineering culture and practice that seeks to unify software development and operations. It integrates development, operations, and quality assurance to strongly advocate for automation throughout the software development process. The ultimate goal of DevOps is to enhance communication and collaboration between different business units within an organization to produce high-quality products. DevOps provides various tools such as Git, Jenkins, and Selenium to automate the entire workflow, improving productivity and delivering the product faster. DevOps optimizes development and operational activities through structured processes, collaboration, and automation, managing end-to-end engineering processes. By promoting increased communication and collaboration, DevOps enables organizations to deliver products to customers within a short time, ensuring total customer satisfaction. DevOps is a culture change that extends beyond a simple tool, affecting paradigms in automation, measuring risks, and facilitating ease of sharing. [11] Simply put, DevOps aligns development and IT operations to improve communication and collaboration.

Why is DevOps Needed?

DevOps is needed because traditional software development methods often result in lengthy development cycles, frequent deployment failures, and inefficient communication between development and operations teams. DevOps addresses these issues by integrating development, operations, and quality assurance into a single, collaborative team.

DevOps lowers the risk of human error and helps businesses to launch applications more quickly and with greater quality by promoting automation and standardization throughout the whole software development lifecycle. DevOps also promotes continuous delivery and continuous integration, allowing teams to frequently release software updates and address issues quickly. It promotes a culture of cooperation and communication that enables various teams to collaborate easily, recognize issues early on, and find solutions quickly. This leads to better alignment of business goals and technology, resulting in a better understanding of customer needs and more satisfied customers.

DevOps is expected to support effort and communication across different teams inside an organization as well as to expand the effectiveness, quality, and speed of software development and delivery.

How Is DevOps Different From Traditional IT?

DevOps is different from traditional IT in several ways:

Siloed vs Collaborative Approach: Traditional IT organizations are often siloed, with developers, operations, and quality assurance teams working independently of one another, on the other hand DevOps encourages a collaborative approach where all the teams in an organization work together to accomplish a common objective.

Manual vs Automated Processes: Traditional IT relies heavily on manual processes, which can be slow and prone to error. DevOps emphasizes automation, which speeds up development and deployment, improves accuracy, and reduces the risk of human error.

Long vs Short Development Cycles: Traditional IT often involves long development cycles, with software releases happening every few months or even years. In contrast, DevOps promotes short development cycles, with releases happening frequently (sometimes multiple times per day).

Reactive vs Proactive Problem Solving: Traditional IT tends to be reactive, with teams waiting for problems to occur before taking action. DevOps promotes a proactive approach, with teams continuously monitoring systems and looking for opportunities to improve processes and prevent problems from occurring in the first place.

Limited vs Comprehensive View: Traditional IT often has a limited view of the entire software development and delivery process, with different teams focused only on their own areas of responsibility. With all teams collaborating to deliver high-quality software to clients, DevOps offers a more thorough perspective of the entire process.

DevOps emphasizes cooperation, automation, continuous improvement, and a broad perspective of the entire process, and it marks a substantial shift in how IT firms approach software development and delivery.

Why Is Devops Used?

DevOps enables Agile Development Teams to achieve Continuous Integration and Continuous Delivery, which accelerates the release of products to the market. However, as it became evident, optimizing continuous integration alone does not make the entire software delivery lifecycle efficient, nor does it drive organizational efficiency. For maximum efficiency, all components of the software delivery lifecycle must function seamlessly, like a well-oiled machine. This problem is addressed by DevOps, which emphasizes automation and standardization across the software development process and encourages cooperation and communication among the development, operations, and quality assurance teams. [12] Other Important reasons are:

Predictability: DevOps enables a significantly lower failure rate of new releases, resulting in a more predictable software delivery process. Reproducibility: By versioning everything, DevOps allows for easy restoration of earlier versions when needed, enhancing reproducibility of the software.

Maintainability: DevOps promotes an effortless recovery process in the event of a new release crashing or disabling the current system, improving maintainability.

Time to market: DevOps streamlines software delivery, reducing the time to market by up to 50%, particularly for digital and mobile applications.

Greater Quality: DevOps helps teams provide improved quality of application development by incorporating infrastructure issues, resulting in a higher quality end product.

Reduced Risk: DevOps incorporates security aspects into the software delivery lifecycle, reducing defects across the lifecycle and resulting in reduced risk.

Resiliency: DevOps promotes a more stable, secure, and auditable operational state for the software system, enhancing resiliency.

Cost Efficiency: DevOps offers cost efficiency in the software development process, aligning with the aspiration of IT companies' management to reduce costs.

Modular Code Development: DevOps, based on agile programming methods, allows breaking larger code bases into smaller, more manageable chunks, promoting modular code development.

DevOps offers a range of benefits, including increased predictability, reproducibility, maintainability, time to market, quality, reduced risk, resiliency, cost efficiency, and modular code development.

DevOps Lifecycle

DevOps is a software engineering methodology that combines development, operations, and quality assurance. This method, which supports the quick and dependable delivery of high-quality software and services, includes the DevOps lifecycle. DevOps fosters communication and collaboration between developers and operations teams, sharing tasks and responsibilities and empowering teams with full accountability for their service and its underlying technology stack. [1]

The Continuous DevOps lifecycle consists of several stages. In the Development stage, software is developed constantly, and the entire process is divided into small development cycles to speed up the software development and delivery process. The Testing stage involves the use of tools like Selenium to identify and fix bugs in the new piece of code. In the Integration stage, new functionality is integrated with the existing code, and continuous integration and testing are conducted. The deployment procedure runs constantly during the deployment stage, ensuring that any modifications to the code have no negative effects on the performance of high-traffic websites. Finally, in the Monitoring stage, the operations team takes care of any inappropriate system behavior or bugs found in production.

DevOps Principles

A software development methodology called DevOps attempts to enhance teamwork and communication between the development and operations teams. Here are the six principles that are essential for adopting DevOps:

Customer-Centric Action: Customer-centric activity is required of DevOps teams, which entails continuous investment in goods and services to satisfy consumers' expectations. This principle focuses on understanding customer requirements and feedback to drive product development.

End-To-End Responsibility: DevOps teams should expect responsibility for the functionality of the product or administration across each phase of its lifecycle, from design to deployment and maintenance. This approach makes sure that the team is responsible for the product's quality and the users' pleasure.

Continuous Improvement: To reduce waste and hasten the improvement of given products or services, DevOps culture places a strong emphasis on continuous improvement. This principle encourages DevOps teams to continuously evaluate and improve their processes, tools, and methodologies.

Automate Everything: The DevOps method, which tries to automate as much of the software development and deployment process as possible, is founded on the essential notion of automation. This includes automation of testing, building, and deploying software, as well as automating infrastructure management tasks.

Work As One Team: The roles of the designer, developer, and tester are predefined in the DevOps culture. All they need to do is work as one team with complete collaboration, sharing information and feedback, and breaking down silos between departments.

DevOps is expected to support, test and communicate across different teams inside an organization and use the results of these tests to expand the effectiveness, quality, and speed of software development and delivery. This includes monitoring the production environment to quickly identify and fix any issues that arise.

Methodology

The Figure 1 illustrates the complete cycle of release management in DevOps. The initial input consists of the source code that must be created offline using Visual Studio. This source code is then copied onto the online Visual Studio, and a build pipeline is created. A build is a piece of software that the development team has produced and sent to the software testing team to test. It represents a single execution of a pipeline and includes the logs and test results associated with the pipeline. Artifacts, which are static files that cannot be modified, are generated after the creation of the build. An artifact is a collection of files or packages that are published by a build and are subsequently made available to other tasks such as distribution or deployment. Once the artifacts have been generated, the project is ready for release, and the desired output is obtained. A release is the execution of a release pipeline and comprises deployments to multiple stages. A deployment target can be a virtual machine, web app, container, or any other service that hosts the application being developed. The pipeline can deploy the application to one or more deployment targets following the conclusion of the build and the execution of the tests**.**

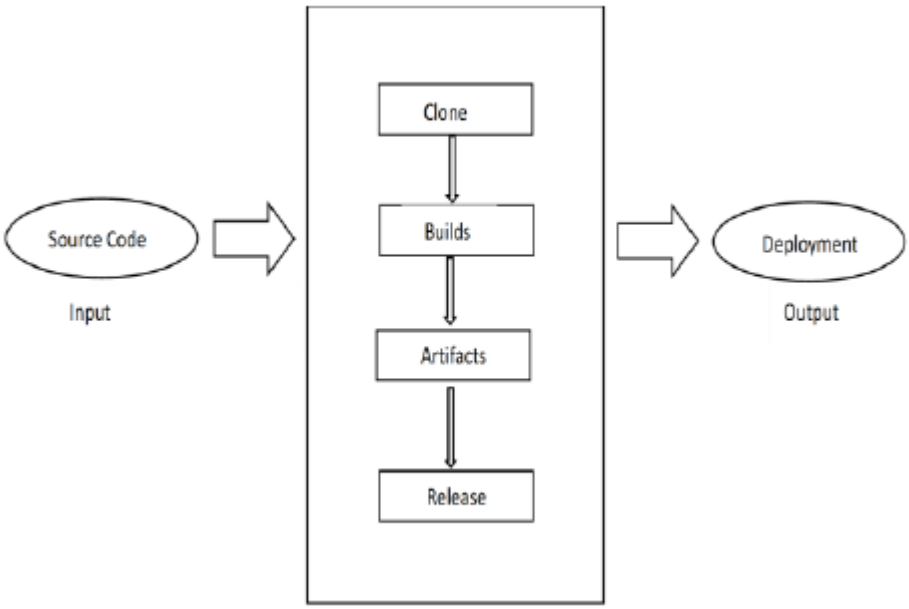

Fig. 1 - Release Management Cycle

Survey:

1. GIT: Git is a popular version control system with several advantages, such as distributed development, ease of branching and merging, faster performance, and support for both small and large-scale projects. Without the need for a centralized server, Git's distributed architecture enables better collaboration amongst multiple engineers working on the same codebase.

Git's branching and merging features also make it simple to work on several features or bug fixes at once without interfering with one another's progress. Git does have significant drawbacks, though, including a high learning curve, the requirement for command-line expertise, and the possibility of conflicts and errors while merging changes. Additionally, Git can be complex to use in larger teams or complex projects, and there may be compatibility issues with some tools or platforms. Git is still a well-liked and widely-used version control system with lots of advantages, but not all projects or teams may be the ideal candidates for it.

2. JENKINS: Development teams utilize Jenkins, a well-liked open-source continuous integration and continuous delivery technology, to automate their software development process. Jenkins has a number of benefits, such as flexibility, simplicity of installation, and a large selection of plugins that enable modification to meet the demands of particular projects. Moreover, Jenkins has a simple to-utilize UI that helps developers manage and monitor builds. However, managing a lot of plugins can cause stability problems as Jenkins can be resource-intensive. Jenkins' complexity may also make it difficult to set up and manage for less experienced users.

3. CHEF: Infrastructure administration is automated using the configuration management tool known as Chef. One of Chef's benefits is its capacity to automate difficult infrastructure management activities. Other benefits include increased scalability and improved consistency across several servers. Additionally, it provides the option to version control infrastructure as code, lowering the possibility of human error and facilitating simple rollbacks. Chef has a severe learning curve, nevertheless, and requires substantial familiarity with its domain-specific vocabulary. The initial setup might be time-consuming, and configuring for more complicated infrastructure configurations can be difficult. Additionally, some users have reported problems with plugin management and version compatibility.

4. PUPPET: Infrastructure as code management is made possible by the open-source configuration management tool puppet. It has the capacity to automate routine processes, enforce configuration and consistency, and offer a centralized control system for managing infrastructure, among other benefits. Additionally, it has a sizable and vibrant user base that actively participates in its growth and exchanges knowledge and experience. Puppet's steep learning curve, which can make it challenging for newcomers to get started, as well as its complicated and occasionally constrictive DSL syntax, are some of its drawbacks, too. The plenty of modules and configuration option in Puppet ecosystem can also be confounding, making it difficult to choose the best alternative for a certain use scenario.

5. SELENIUM: A popular open-source tool for test automation, Selenium has both benefits and drawbacks. The support for different programming languages, browser interoperability, cross-platform support, and a sizable community for assistance and resources are some of its main attractions. Parallel test execution is also supported by Selenium, enabling quicker testing and feedback. Its drawbacks, however, include the demand for technical know-how, the need for regular and dependable test environments, and the absence of integrated reporting and analytics functionalities. In addition, Selenium can be time-consuming to install and maintain, and its tests sometimes need frequent upgrades.

Conclusion

The overall goal of DevOps is to speed the entire software development process by integrating development, operations, and quality control. DevOps focuses on automation and monitoring in order to produce high-quality goods more quickly. Without a backup mechanism, the version control functionality enables simple rollback to earlier versions of the code. The quality of software products can be considerably increased by using DevOps with a focus on culture, automation, measurement, and sharing, according to research. Companies may improve their testing procedures and eventually provide better products to their customers by correctly applying these strategies.

Reference

[1] Pulasthi Perera. Roshali Silva. Indika perera, Improve software quality through practicing DevOps, 2017 IEEE. 2017 Seventeenth International Conference on Advances in ICT for Emerging Regions (ICTer).

[2] P. Perera, I. Perera, M. Bandara, "Evaluating the Impact of DevOps Practice in Sri Lankan Software Development Organizations", ICT for Emerging Regions, pp. 281-287, 2016.

[3] "Agile Manifesto", Agile Manifesto, 04 2017, Available: http://www.agilemanifesto.org/.

[4] M. Huttermann, "Beginning DevOps for Developers", DevOps for Developers, pp. 3-13, 2012.

[5] D. L. Farroha, B. S. Farroha, "A Framework for Managing Mission Needs Compliance and Trust in the DevOps Environment", Military communicationsConference, pp. 288-293, 2014.

[6] "2016 State of DevOps", Puppet and Dora, 2016.

[7] D. Farley, J. Humble, Continuous Delivery: Reliable Software Releases through Build Test and Deployment Automation, Pearson Education, 2010.

[8] diazhilterscheid, 06 2016, [online] Available: hilterscheid.de/en/ready-for-cams-the-foundation-of-devops/.

[9] "ISO/IEC 9126–1:2001 - Software engineering - Product quality - Part 1: Quality model", ISO, 03 2001, [online] Available: https://www.iso.org/standard/22749.html.

[10] M Rajkumar. Anil Kumar Pole. Prabal anta. Vittalraya Shenoy Adige, DevOps culture and its impact on cloud delivery and software development, 2016 IEEE.

[11] Manish Virmani, Understanding DevOps & bridging the gap from continuous integration to continuous delivery. 03 August 2015,IEEE.