Matthew Trinh

CSD 380

10/27/2024

The history of DevOps

The collaboration of development and IT operations is gradually becoming the essence of any business. The concept of DevOps is continuously evolving. DevOps is not just a technological shift in software development; instead, it brings a tangible shift in the overall culture of an organization by challenging and inculcating the best practices followed in the software development segment and IT operations. The power of DevOps is gradually motivating organizations to adopt this transformative journey to streamline their operational collaborations, which helps accelerate their product-to-market journey and propel the overall business. This text encapsulates the history and evolution of DevOps. It also walks through a journey that helps prospective readers understand the impact, strategies, and laws of DevOps through the eyes of pioneers and practitioners. In addition, it emphasizes the strength of cultural transformation and evolution, which is necessary for the DevOps journey.

The key to the growing impact of DevOps is continuous improvements that help any organization cater to the needs and satisfaction of the customers. It is very important for any organization to collaborate with their business partners or customers to capture the consensus. This text will help the readers understand how to apply the ideal blends of practices and what good looks like, encouraging evolution and continuous improvement to steer their DevOps journey. Readers will realize that the quest for DevOps is not only about adoption but a journey of winning the hearts and minds of their peers. It sheds light on the exhaustive exploration that elucidates the core and thrust of DevOps evident in software engineering and IT operations. It emphasizes building, deploying, and testing code, which is a related activity of DevOps software that is the foundation and is driven by ideal quality fundamental practices such as code automation, configuration management, continuous integration, continuous deployment, automation testing, containers, monitoring, and log management. The essence of IT notably lies in leveraging time, money, and effort to optimize this cycle for expedited delivery, delivery repeatability, and high performance.

The principles of Lean Engineering have been strongly influential in the modern DevOps movement that we know today. As advocated by the foremost thought leaders of the Agile and Lean movements, cross-functional collaboration is at the heart of Lean Engineering. Input is collected at all stages of the process, and continuous improvement is expected. The differences between Agile and Lean often revolve around processes and project management. Kanban, for example, which comes from Lean thinking, can be thought of as a more flexible issue-tracking system. Agile, however, defines a 'sprint' which is not supposed to change within a two-week time frame. Large-scale Agile practices are likewise less focused on overall value streams than a Lean transformation across a company. Whereas some may view DevOps as a natural evolution of Agile, others may feel that the evolution comes from manufacturing. Concepts such as continuous improvement and is a pillar of Lean thinking, can be found in the history of contemporary DevOps stories that started in the finance industry. In the same way that continuous deployment is often cited as a key trait of DevOps, Lean is famous for concepts such as conveyor lines and single-piece flow. Only recently were we, as an industry, capable of highly iterative development processes due to modern cloud infrastructure tools.

The origins of DevOps can be traced back to Agile, Lean, Systems Thinking, and the Theory of Constraints. In some ways, DevOps formalized ideas that were floating around in the community revolving around lean principles and the elimination of waste in application development. In the Toyota Production System, seven types of waste (non-value adding processes) are discussed, namely overproduction, waiting time, unnecessary conveyance, over-processing or incorrect processing, correction (make-up for mistakes), repeated operation, and inefficient motion. The first three of these wastes, overproduction, waiting time, and unnecessary conveyance, apply in the domain of software development. It was observed that typical application development processes were guilty of producing large amounts of unnecessary or low-value artifacts, waiting and hand-holding them between different process steps, and unnecessarily moving them between teams and geographical locations. These generated a collective group of principles that facilitated a systematic examination of processes where work was described, planned, prioritized, delivered, verified, and validated, namely: Agile, Lean IT, Systems Thinking, and the Theory of Constraints. Today, DevOps is responsible for these coalescing principles in the areas of agility, scale, quality, security, and resilience, making the DevOps space unique and rich with cultural conversations, technology choices, and organizational challenges.

One of the first attempts to formulate the DevOps idea started to bring together lots of ideas that have been floating around in the community revolving around lean principles and the elimination of waste in application development. Agile is a philosophy and a collection of principles and practices for software development that offers the same.

Another early approach to formalize the DevOps concepts was proposed. It maintains that to manage Site Reliability and Service Availability, the Agile methodology can be applicable not just for development; in the work with the operations team regarding its activities: Agile methodology can be applied in the task of the operations team and since it is not only for a scripting issue. DevOps is a cultural movement, and many tools can be used to support a significant change in collaboration between development and operations. Agile works under a different alignment in this movement, and to explain this, a presentation is prepared containing an overview of Agile and shows how it can also be used in support of DevOps.

Research finds it substantially easier for DevOps teams to respond to customer feedback and complete deliveries quickly and at low cost. The ability to respond quickly to customer feedback leads to increased innovation. Specifically, subjects in DevOps teams experiment with new ideas, new technologies, and new tools significantly more. They also report they are more eager to experiment with new ideas compared to their counterparts. Furthermore, after initially adopting DevOps, teams tend to undergo some form of change, including increased levels of quality assurance, human collaboration, integration of security best practices, collaboration from neighboring teams, program management, and risk management. These transition changes reflect an adaptation of the DevOps practices in response to the unique demands and characteristics of the team or increased understanding of DevOps principles, indicating the maturity or longevity of the DevOps principles within the organization.

Underlying all organizational ICT work activity, there are five themes: the speed of work, information and communication, reflexivity, scalar orientation, and the particular, universal, general relationship. Speed of work is concerned with the urgency or hurry attached to work tasks. The greater the speed, the more the behavior of the individual or team can be characterized as 'service to information rather than production of information.' Development organizations typically engage in rapid response to customer demand, whether through either realization of a new software artifact or the re-deployment of an existing, working piece of software. Speed of work in such context is paramount. DevOps, defined as a set of practices intended to reduce the time between committing a change to a system and the change being placed into normal production and thereby increasing the frequency of code releases, focuses primarily on the timely delivery of software service artifacts, such as web applications shared for public consumption. Speed of work, interconnected with the other themes, is central to achieving the primary purpose of a DevOps team. As a consequence, DevOps changes how organizations prioritize the multi-level projects seen in the context of dynamic capabilities and the functions of the organization within the marketplace.

The Agile Manifesto consists of four important values and twelve principles. It guides Agile methodologies and provides rules and most often needed guidelines. The four core values of the Agile Manifesto are: Individuals and interactions over processes and tools. Working software over comprehensive documentation Customer collaboration over contract negotiation Responding to change over following a plan After declaring these core values, the manifesto offers the twelve principles as guidelines to follow: Our highest priority is to satisfy the customer through early and continuous delivery of valuable software. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference for the shorter timescale. Business people and developers must work together daily throughout the project. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation. Working software is the primary measure of progress. Agile processes promote sustainable development: the sponsors, developers, and users should be able to maintain a constant pace indefinitely. Simplicity—the art of maximizing the amount of work not done—is essential. The best architectures, requirements, and designs emerge from self-organizing teams. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly. The Agile Manifesto also includes thirteen "not-so-important" aspects in the form of a "Statement of Commonality."

DevOps is a complex and multifaceted phenomenon that comprises many diverse elements. Its very polycentricity in everyday practice allows it to constantly evolve through co-creation and reifications of different concepts. However, it is still important to keep DevOps broad enough to maintain a complementary portfolio to Agile and Lean. The central elements of DevOps should, therefore, always stay based on the listed key principles. Not all activities make use of each key feature, but philosophical "DevOps thinking" should be based on all these suggested elements. This list will evolve as DevOps evolves.

There are two additional key principles for DevOps that ensure co-creation between development ("Dev") and operations ("Ops") along with acceptance in IT change processes. The idea of DevOps is to improve cooperation and derive common ground. The idea is not to unify. The key to effectiveness for DevOps change requires automation in many or all processes. This allows improved speed and shorter lead times, along with effective and efficient delivery. The entire organization moves, a change is made every day (or week), and change is a fact of life.

The natural relationship that exists between Agile methodologies and CI and automation underlies the strong integration between DevOps culture and Agile culture. In fact, many people believe that DevOps practices are nothing more than the process of applying Agile techniques to the operations area of software development, since both involve linear, initiative-based, and closely interrelated methods. Agile proposes a consistent methodology that maximizes the collaboration of all phases of software development. DevOps represents the use of strategic thinking and actions to apply the culture and practices inherent in Agile, between the developers, the operations, and the functions that support the process, with a framework that brings innovation, reliability, predictability, and quality. However, aligning Agile methodologies with DevOps, and extracting the best that each one has to offer, is an extremely challenging task, but that should be pursued at all costs. The radical search for innovation is a characteristic common to both the working methodologies of the Agile area and the leveraging gains methodology, as well as the property of reducing costs, delivery times, risks, unpredictability, quality problems, and communication difficulties between the participating teams. Agile’s ability to integrate different cultures among its participants may be a means of complementing the relationships existing or that must be established through the services provided by a maintenance team and a development team, and even by the negotiations existing between the teams of a software production environment and the teams of desktop software, mobile, or cloud services. DevOps practices in an Agile environment are the key link to achieving sustainability for this relationship.

The Continuous Delivery movement marks a new turning point in software engineering, addressing yet another problem, the first being development and testing, while achieving software delivery at a faster rate. This movement not only develops and protects high-value software but also releases software reliably and efficiently. In the domain where it becomes important to achieve time, quality, and frequency, the word 'time' has emerged again. In a nutshell, the movement refactors every software development project into a state suitable for world-class delivery. To respond to this pressing issue and attain peace in the development community, the movement seems to focus on changing this mindset. Valuable projects with tested code have made significant progress. A major contributor to this movement is the development of a new architecture, key techniques, and systemic thinking patterns to significantly increase the success rates of the continuous delivery process. The technologies that enable the movement have been combined to form an approach for the systemic implementation of the continuous deployment pipeline, allowing organizations to make rapid, low-risk releases of their software-based products. The practices and processes form a coherent framework across the activities of the value stream with visibility to enable failure detection through a rejection decision. With the increased clarity, this decision can be applied from that point in the organization.

Infrastructure as code defines the state of infrastructure with unique properties. Using code and other associated automation tools, an environment (including servers, networks, and so on) in every authoritative data center is managed in deployment, provisioning, updating, and scaling out the actual free data center. Configuration management is an abbreviation for the state of system actions and maintenance of that state. Obviously, it helps with automation, portability, validation, proper control, and system consistency. There are popular and applied techniques and practices such as Puppet, Chef, DSC, Ansible, SaltStack, and other tools. For a long time, DevOps has been described as configuration management. Two terms should be considered; however, the primary focus of DevOps is the concept of a continuous self-growing archive of capabilities and resources that are bound by central services and systems administration through an "information" method. As soon as practical, we are no longer focusing on this space of "Resources" or "Configuration," and in point of fact, we are not trying to take a few generalizations so far.

There are a number of ways in which the work on Agile can be linked to DevOps. We see a repetition of focus on the people element in approaches like Agile and Lean thinking and DevOps. DevOps practice is not so prescriptive as to how to solve the problems and thereby allows organizations like financial institutions to do DevOps while satisfying the current pertinent regulations. Software quality is a big mandate at financial institutions for tolerance of older and even outdated system infrastructure in order to keep risk appetite in check. Lower levels of regulatory pressure usually see the accelerated adoption of visual management protocols, which should result in a more consistent value realization of DevOps and Agile. DevOps and Agile have valid strategies and tools that are effective in reducing the time to production but are not successful at building code that has spider legs that enable significant reuse. Requirements identification is still best served without a DevOps play. Compliance should be handled separately, traditionally separate and subsequent to DevOps and Agile activities.

This section outlined the history and evolution of DevOps up to the present day. Through our examination, we found that DevOps and its related models have been developing for over 15 years, going through multiple phases with a focus on continuous improvement. Originally, DevOps was a collaborative model but has since transformed into a continuous improvement mechanism supported by various tools. The key elements and mechanisms of DevOps have evolved significantly over time, gaining public recognition and accumulating industry experiences and insights. These experiences have contributed to the establishment and spread of the theory and practice of DevOps. However, there is still a need for further efforts to enhance the systematization of DevOps maturity models and advance DevOps theories. Currently, DevOps models and methods are largely based on the Agile concept, but there is a growing presence of diverse development properties such as AI-driven, IoT, data-driven, and 5G. Consequently, DevOps will continue to evolve into its next phase gradually.