International Journal of Research in Advanced Engineering and Technology
www.allengineeringjournal.in
ISSN: 2455-0876
Received: 06-04-2024, Accepted: 29-04-2024, Published: 21-05-2024
Volume 10, Issue 1, 2024, Page No. 27-29

Navigating the DevOps landscape: Insights and perspectives

Rahaman Nagiur, Perfilev Dmitrii
Department of Computer Engineering, Lanzhou University of Technology, China

Abstract
DevOps is the integration of duties done by a company's application development and systems operations teams, and its implementation has been seen in a variety of ways. As a significant innovation in Information System development, DevOps embodies an operational ideology to enhance communication between development and operations, particularly as more operational aspects become programmable. The overarching objective of the DevOps approach is to streamline the application development and deployment process, aligning the efforts of development quality assurance (QA) and operations teams. This approach involves redistributing specific responsibilities from operations to development teams, facilitating continuous development, integration, delivery, and monitoring processes. The imperative has accelerated the need to dismantle the traditional silos between development and operations to release code faster and more frequently, enabling organizations to respond in a more agile manner to evolving business requirements. Furthermore, the increasing adoption of cloud computing, advancements in software-defined infrastructures, microservices, containers, and automation have also contributed to the drive for breaking down these silos. This abstract highlights the essence of DevOps, its objectives, and the contextual factors motivating its adoption, providing a foundation for further exploration and understanding in the field.

Keywords: DevOps, operations, software engineering, methodology

Introduction

One of the most well-liked approaches for bridging the gap between software development and operations in recent years is DevOps. The DevOps technique calls for collaboration between development teams and operational professionals to quickly produce software to deploy IT solutions to the market. According to research, the fundamental goal of DevOps is to install new features efficiently and reduce the time it takes from development to production. These data demonstrate the importance of this factor in enhancing software development outcomes. However, DevOps is more than just a technique; it is a collaborative and automated culture that helps to prevent burnout and increase productivity. Over 36,000 experts worldwide participated in research on DevOps methods, promoting effective software delivery and operational efficiency. In particular, automation lowers the amount of errors and ensures code quality. A thorough literature evaluation also highlights the importance of including security in the DevOps process to guarantee secure development and efficient product delivery. This demonstrates how modern organizations in the digital transformation age confront new problems requiring IT staff to adapt to changing technology and embrace customer-centric methods. As a result, classic software development models such as the Waterfall Model and Spiral Model have been replaced with agile approaches like Kanban and Scrum. DevOps has emerged as the most relevant approach to software development, resulting in a surge in demand for specific IT solutions. Rapid software deployment is no longer simply a competitive advantage but a need in today's industry. The need for DevOps professionals has gradually increased over the last few years, and this trend is projected to continue. As demand has increased, so have the duties of DevOps engineers. They now work with developers, testers, and system administrators, automating processes and configuring infrastructure. DevOps engineers are in great demand in the job market, but existing training programs may not necessarily offer them the most current knowledge and abilities.

DevOps

DevOps is a software development strategy founded on agile and lean concepts that encourages communication among business owners, development teams, operations teams, and quality assurance teams. Its primary goal is to enable the continual and steady distribution of software.
Furthermore, DevOps fosters cross-functional cooperation, shared ownership of business activities, and a commitment to collective goals. It is distinguished by a culture that fosters positive interpersonal ties inside the organization.
In essence, DevOps is a movement that aims to improve the agility of IT service delivery. It refers to methods that assist software's speedy and dependable delivery.

DevOps Principles

Several key concepts guide the DevOps approach to software development and operations. These principles include:

1. **Automation:** Automation is vital in DevOps, aiming to reduce manual, repetitive tasks and streamline processes. Organizations can achieve greater efficiency, minimize errors, and accelerate their development cycles by automating various aspects of software development, deployment, testing, and infrastructure management. Automation enables teams to focus on higher-value activities while maintaining consistent and reliable processes throughout the software life cycle.

2. **Collaboration:** Collaboration is at the core of DevOps, emphasizing the need for cross-functional teams to work together seamlessly. Developers, operations personnel, and other stakeholders collaborate closely throughout the software development process. DevOps

teams may accelerate innovation by building a culture of shared accountability and good communication. Collaboration enables faster problem-solving and decision-making, leading to efficient and successful project outcomes.

3. **Continuous Integration and Delivery:** DevOps promotes the practice of continuous integration and delivery (CI/CD) to streamline the software release process. CI involves regularly merging code changes into a shared repository, where automated tests are performed to detect integration issues early. The CD focuses on automating the deployment process, allowing frequent and reliable software releases. CI/CD enables teams to deliver software updates rapidly, frequently, and consistently, reducing lead times and ensuring a smooth and reliable deployment process.

4. **Feedback Loops:** DevOps emphasizes the importance of obtaining feedback from various stakeholders throughout the software development life-cycle. Feedback loops involve engaging with customers, end-users, and other relevant parties to gather insights, validate assumptions, and refine software solutions. By establishing short feedback loops, DevOps teams can quickly respond to changing requirements, identify areas for improvement, and align their products with customer needs. Continuous feedback drives iterative development enhances product quality and fosters customer satisfaction.

5. **Infrastructure as Code:** Infrastructure as Code (IaC) is a fundamental DevOps principle that advocates for programmatically defining and managing infrastructure resources. IaC treats infrastructure components like servers, networks, and databases as code. This approach enables teams to provision, configure, and manage infrastructure resources through version-controlled and automated processes. IaC promotes consistency, scalability, and repeatability in infrastructure management, allowing for agile and efficient deployment and scaling of applications.

DevOps life-cycle

DevOps is designed to be a cross-functional method of working; hence, instead of a single DevOps tool, there are sets (or "toolchains") of numerous tools. Such DevOps tools are expected to fall into one or more categories, reflecting essential parts of the development and delivery processes. DevOps is the deep integration of development and operations. Understanding DevOps requires an understanding of the DevOps life cycle.

1. Development

In the DevOps stage, software development is continuous. This step involves breaking down the development process into smaller cycles. This enables the DevOps team to speed up the software development and delivery. Continuous development requires developing and producing several versions of the code using SVN and Git and finally packaging the code into an executable folder for distribution to quality analysts for testing.

2. Testing

Testing is a process that occurs concurrently with the actual use of the application in DevOps. Beta testers can produce results while assuring the application can be used as intended in a live setting. Testing gives additional information about various program parts, which can be fed back into the development process to help enhance the product. Quality Analysts utilize tools such as Selenium and Junit to test software for faults and ensure its functionality.

3. Integration

Continuous integration is integrating new features into existing code and testing it. Constant development is only possible through continuous integration and testing. The integration process contains several processes, such as preparing the numerous tests performed in the following phase and determining whether the produced code can perform the desired actions as required. Jenkins is an integration tool that triggers any changes made to the code, either automatically or manually.

4. Deployment

The deployment process runs continually. It is done so that any changes made to the code at any time will not interfere with the operation of a heavy-traffic website. Automation tools such as SaltStack, Puppet, and Chef play a significant role.

5. Monitoring

The operational phase in DevOps is the monitoring phase; during this phase, the operations team will address any incorrect system behavior or problems discovered in production. The team employs well-known tools such as Sensu, NewRelic, and Nagios.

What makes DevOps significant

DevOps is significant because it addresses the challenges and inefficiencies that arise when there is a lack of collaboration and coordination between different phases of software development, namely development, testing, and deployment.

Traditionally, software developers would spend weeks or months writing code, which would then be handed over to the Quality Assurance (QA) team for testing. Once the code passes QA, it is deployed to the operations team. This sequential handover process often led to delays and a need for more communication between teams.

In this traditional model, developers would write the code and pass it on to the deployment team. If any issues arose during deployment, the deployment team would either try to fix them or send the code back to the development team for bug fixes. This back-and-forth process further slowed down the software development process.

DevOps aims to bridge the gap between development, testing, and deployment by fostering collaboration and communication among these teams. It promotes a cultural shift towards shared responsibility and accountability for the entire software lifecycle. Developers can quickly deliver changes by adopting DevOps practices and principles, while operations teams can ensure reliability and stability.

Handling Barriers and Supporting Collaboration in DevOps Adoption

1. Development teams primarily focus on creating software that meets functional and quality

requirements, aiming to facilitate effective changes. Conversely, Operations teams prioritize operability aspects such as availability, stability, service improvement, and cost efficiency.
2. It is crucial to anticipate significant resistance, particularly from individuals who need help to grasp the potential benefits of adopting DevOps practices. Such resistance often stems from an unwillingness to change established attitudes and behaviors.
3. Breaking down barriers across the extended Development and Operations organization necessitates various changes, including the reshuffling and relocating team members. These modifications are essential to foster collaboration and a shared mindset.
4. When attempting to instill a DevOps mindset within development and operations teams, it is not uncommon to encounter resistance from existing functional silos. Overcoming this resistance requires careful navigation and effective communication.
5. Individuals unaccustomed to working in cohesive, cross-functional teams may need help adapting to the new operational paradigm introduced by DevOps. Adapting to this new normal requires a shift in mindset and embracing collaborative work practices.
6. Team members may need more shared terminology to face communication challenges. It is vital to establish clear and consistent communication channels, along with promoting the use of common terminology and concepts, to mitigate such challenges.

Within the framework of the DevOps model, the traditional isolation of development, operations, and testing teams is dissolved. These teams are often merged into a cohesive unit, where engineers collaborate across the entire application lifecycle. This integrated approach enables the development of a diverse skill set that extends beyond individual functions. Additionally, integrating security teams becomes more seamless throughout the application lifecycle.

DevOps dismantles the barriers between development and operations teams, fostering a unified approach that leads to improved and expedited outcomes. By doing so, it effectively addresses communication challenges and encourages cohesive collaboration. In this collaborative environment, neither the developers nor the operations team resort to blame when issues arise.

Furthermore, DevOps facilitates the seamless adoption of new technologies during software development, enabling developers to embrace changes without encountering resistance.

Conclusion

In this review research, we have provided a concise overview of the introduction, evolution, necessity, and diverse manifestations of DevOps. The study has explored the knowledge landscape and identified areas for further long-term research in DevOps. It has shed light on the advantages and challenges of adopting DevOps, including increased release frequency and enhanced automation. Furthermore, it emphasizes the importance of leveraging a divide-and-conquer strategy by distributing expertise across development and operations teams. This comprehensive review report contributes to the broader understanding of DevOps, offering valuable insights for practitioners and researchers alike. Its findings have implications on a large scale, impacting the way DevOps is practiced and understood in various contexts.

References

1. Luz WP, Pinto G, Bonifácio R. Adopting DevOps in the real world: A theory, a model, and a case study. Journal of Systems and Software,2019:157:110384. doi: 10.1016/j.jss.2019.07.083.
2. Wedel F. Devops: A Systematic Literature Review,2019, 16.
3. Jha P, Khan R. A Review Paper on DevOps: Beginning and More To Know. IJCA,2018:180(48):16-20. doi: 10.5120/ijca2018917253.
4. Mishra A, Otaiwi Z. DevOps and software quality: A systematic mapping. Computer Science Review,2020:38:100308. doi: 10.1016/j.cosrev.2020.100308.
5. Erich FMA, Amrit C, Daneva M. A qualitative study of DevOps usage in practice. J Softw Evol Proc,2017:29(6). doi: 10.1002/smr.1885.
6. Sánchez-Gordón M, Colomo-Palacios R. Characterizing DevOps Culture: A Systematic Literature Review. In: Stamelos I, O'Connor RV, Rout T, Dorling A, editors. Software Process Improvement and Capability Determination. vol. 918. Cham: Springer International Publishing, 2018, 3-15. doi: 10.1007/978-3-030-00623-5_1.
7. Senapathi M, Buchan J, Osman H. DevOps Capabilities, Practices, and Challenges: Insights from a Case Study. In: Proceedings of the 22nd International Conference on Evaluation and Assessment in Software Engineering 2018. Christchurch New Zealand, 2018, 57-67. doi: 10.1145/3210459.3210465.
8. Lwakatare LE, Karvonen T, Bosch J, Olsson HH, Saa S, Kuvaja P. DevOps in practice: A multiple case study of five companies. Information and Software Technology,2019:114:217-230. doi: 10.1016/j.infsof.2019.06.010.
9. Kitchenham B, Charters S. Guidelines for performing Systematic Literature Reviews in Software Engineering. In: Shepperd M, Brito e Abreu F, Rodrigues da Silva A, Pérez-Castillo R, editors. Technology. vol. 1266. Cham: Springer International Publishing, 2020, 184-198. doi: 10.1007/978-3-030-58793-2_15.
10. Jabbari R, bin Ali N, Petersen K, Tanveer B. What is DevOps?: A Systematic Mapping Study on Definitions and Practices. In: Proceedings of the Scientific Workshop Proceedings of XP2016. Edinburgh Scotland UK, 2016, 1-11. doi: 10.1145/2962695.2962707.
11. Sánchez-Gordón M, Colomo-Palacios R. Characterizing DevOps Culture: A Systematic Literature Review. In: Stamelos I, O'Connor RV, Rout T, Dorling A, editors. Software Process Improvement and Capability Determination. vol. 918. Cham: Springer, 2018, 3-15. Available from: https://doi.org/10.1007/978-3-030-00623-5_1.